Spécification formelle d'une application de téléconférence en SDl'92

Mohammed Ouzzif, André Schaff

Rapport interne

Spécification formelle d'une application de téléconférence en SDL'92.

Mohammed Ouzzif et André Schaff

Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA)B.P. 239 54506 VANDOEUVRE-lès-NANCY CEDEX Tél : 03 83 59 20 00 Fax.: 03 83 27 83 19.

Spécification formelle d'une application de téléconférence en SDL'92.

Résumé

Le monde de l'informatique a été marqué ces dernières années par l'avènement du « multimédia » et des «autoroutes de l'information ». L'utilisation conjointe des techniques rapportés à ces deux domaines ont donné naissance à des recherches en informatique : la visioconférence et toutes les activités faisant appel à cette technologie (télé-ingénierie, télé-enseignement, télé-médecine). Ces applications constituent des systèmes complexes et une étape de spécification formelle constitue une phase primordiale pour leurs développements.

Les techniques de description formelle s'avèrent de plus en plus nécessaires ; surtout dans les domaines dont la complexité est assez élevée. En effet ces techniques permettent d'avoir des spécifications claires, exactes, complètes et vérifiables.

Dans ce travail, nous allons présenter une spécification formelle de quelques aspects de coordination dans un système de téléconférence. Cette spécification a été réalisée avec le langage SDL'92 et a été implémentée avec l'outil ObjectGeode. Des simulations de ce système ont été faites grâce à cet outil et différents scénarios ont été vérifiés.

Mots clés : Téléconférence, visioconférence, multimédia, coordination, spécification formelle, technique de description formelle (FDT), SDL'92, ObjectGeode.

Sommaire

I. Introduction

Ces dernières années, le domaine du multimédia a connu une forte évolution. Actuellement, on parle d'un monde de l'information « tout numérique ». Tous les médias, en particulier l'audio et la vidéo, ont pu être intégrés et traités par des ordinateurs. Ceci a été permis grâce à la croissance continue de la puissance des processeurs, l'avènement de nouvelles fonctionnalités des systèmes opératoires, le développement des algorithmes de compression des médias les plus volumineux et l'apparition de nouvelles technologies de stockage (CD-ROM). Ces différents facteurs ont ainsi facilité le développement de puissantes applications multimédias locales.

Par ailleurs, le domaine des réseaux informatiques n'a cessé d'évoluer. Grâce à ce qu'on appelle les « autoroutes de l'information », l'acquisition, l'échange de l'information et le partage de ressources sont devenus des tâches très faciles. Le transport d'autres types d'information tel que l'audio et la vidéo a été rendu possible grâce à la numérisation du monde de la télécommunication et en particulier des réseaux longues distances tel que RNIS *(Réseau Numérique à Intégration de Services)* ainsi que le développement des réseaux à haut débit tel que ATM *(Asyncronous Transfer Mode).*

L'utilisation conjointe des technologies des deux domaines cités plus haut a donné naissance à de nouvelles directions de recherches à savoir la visioconférence et les activités pouvant faire appel à cette technologie. Ces activités s'avèrent multiples, citons le télé-enseignement et la téléconférence [Sass 94].

Ainsi, plusieurs équipes de recherches ont commencé a travailler sur les problèmes liés a ces directions. Cependant, plusieurs d'entre elles se focalisent sur les aspects techniques [Bolo 98] de la visioconférence et peu de travaux s'intéressent à la spécification formelle des applications pouvant être construites au dessus d'un système de visioconférence.

Les techniques de descriptions formelles ont été d'un très grand apport dans le développement des systèmes et applications complexes [Turn 88]. Ces techniques permettent de réaliser des spécifications claires, exactes, complètes et vérifiables. C'est dans ce sens que nous utilisons le langage SDL *(Specification and Description Language)* pour spécifier des aspects liés à un système de téléconférence ainsi que les relations avec son système de visioconférence sous-jacent. Nous avons aussi voulu démontrer, par cette spécification, l'adéquation de ce langage de spécification formelle, a priori destiné aux protocoles de communication, pour la spécification des aspects coordinations dans un système de téléconférence.

Le présent rapport est structuré en cinq sections. La section 2 présente les principaux concepts SDL qui seront utilisés dans notre spécification. La section 3 présente une spécification informelle du système de téléconférence à spécifier. La section 4 sera consacrée à la spécification formelle de ce système en SDL'92. La section 5 donnera une brève description de l'outil ObjectGeode. La section 6 décrit un scénario de cette spécification simulée par ce même outil. Enfin, la section 7 conclura ce travail.

II. Présentation du langage SDL

II.1. Introduction

SDL (*Specification and Description Langage*) est un langage de spécification formelle normalisé par l'IUT-T (*International Union Telecommunication).* Ce langage permet de décrire et de spécifier sans ambiguïté le comportement des systèmes de télécommunication. Les spécifications réalisées avec SDL doivent être formelles dans ce sens qu'il doit être possible de les analyser et de les interpréter sans ambiguïté [IUT-T 93].

Les objectifs généraux qui ont été pris en compte lors de la définition de SDL sont de fournir un langage facile à apprendre, à utiliser et interpréter ; extensible pour permettre un développement ultérieur ; permettant l'application de plusieurs méthodologies de spécification et de conception de système sans supposer a priori une méthodologie particulière quelconque.

Le champ d'application de SDL est la description du comportement des systèmes en temps réel dans certains de leur aspects. Ces applications comprennent :

- le traitement des appels dans les systèmes de communication ;
- la maintenance et le dépannage dans les systèmes de commutation ;
- la commande du système ;
- les fonctions d'exploitation et de maintenance ;
- Les protocoles de communication de données ;
- Les services de télécommunication.

SDL est un langage particulièrement riche, utilisable à la fois pour des spécifications informelles de haut niveau (et/ou formellement incomplètes), des spécifications partiellement formelles et des spécifications détaillées. L'utilisateur doit choisir les parties appropriées de SDL en fonction du niveau de communication souhaité et de l'environnement dans lequel le langage sera utilisé.

Une spécification effectuée avec SDL se présente comme un ensemble de machines d'états communicant en échangeant des messages. Ces machines d'états peuvent aussi communiquer avec l'environnement du système. Elles s'exécutent en parallèle et plusieurs instances du même processus peuvent concurremment exister [TURN 93].

On va présenter dans ce qui suit, quelques concepts du langage SDL. Cette description ne se veut pas exhaustive, mais permettra au lecteur de comprendre la spécification du système de téléconférence présenté au paragraphe III.

II.2. Aspect architectural de SDL

Une spécification écrite en SDL est représentée par l'objet *système* qui constitue l'objet de plus haut niveau hiérarchique d'une description SDL. La création de cet objet implique la création d'une frontière entre le système à modéliser et son *environnement* externe.

L'objectif d'un système SDL est de modéliser un ensemble de machines d'états étendues communicants groupées en *blocs* . Un bloc peut être décomposé à son tour en d'autres blocs. Cette décomposition constitue ainsi une structure hiérarchique arborescente dans une spécification SDL (fig 1).

Un bloc terminal est constitué de *processus*. Ces processus sont décrits par des automates d'états finis et constituent l'unité du comportement dynamique du système. Un processus peut faire usage de *procédures* qui représentent des parties de son comportement.

La figure 1 donne une représentation graphique de la structure hiérarchique que peut avoir une spécification SDL. Dans cette figure, le système s est composé des blocs : Bloc1, Bloc2 et Bloc3. Le bloc Bloc2 est constitué à son tour des deux blocs : Bloc21 et Bloc22. Le bloc Bloc21 est composé des deux processus : Procecessus1 et Processus2. Le processus Processus1 fait appel à deux procédures : Procedure1 et Procedure2.

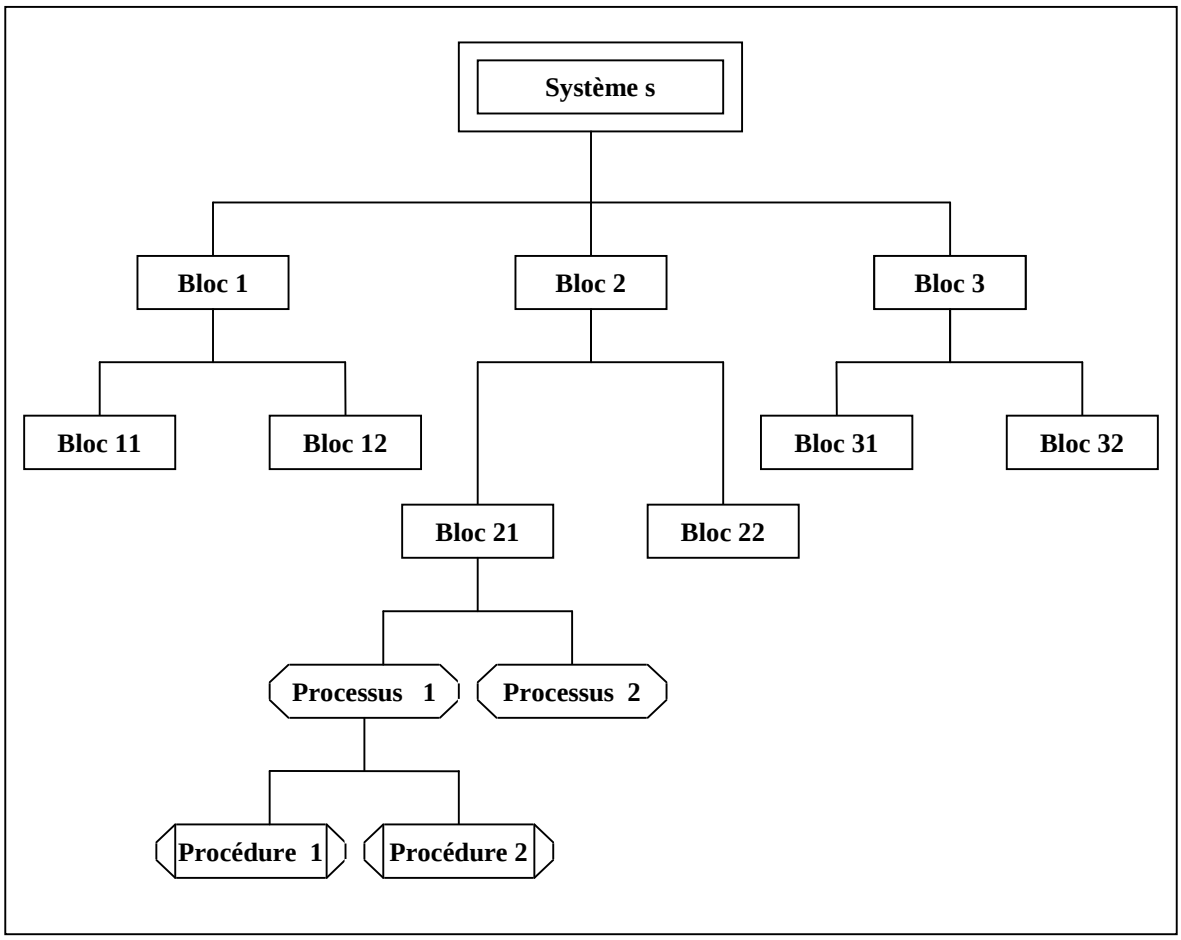


Figure 1. **Structure hiérarchique d'une spécification SDL.**

II.3. Aspect communication dans SDL

Les objets SDL communiquent par envoi et réception de *signaux* qui transportent des messages. Les messages peuvent être des valeurs, litéral ou expression, qui sont affectées à des variables des processus lors de la réception des signaux. Ces signaux sont transportés via des *canaux* ou des *routes*.

Les canaux permettent l'échange des signaux entre deux blocs et entre un bloc et son environnement externe. La figure 2 matérialise une communication de ce type entre les deux blocs « a » et « b » du système s via le canal « d ». Ce canal est unidirectionnel et permet le transport des signaux « p » et « t » du bloc « a » vers le bloc « b ». Le bloc « a »

peut recevoir le signal «p ». On peut aussi avoir des canaux bidirectionnels permettant l'échange des signaux dans les deux sens entre deux blocs.

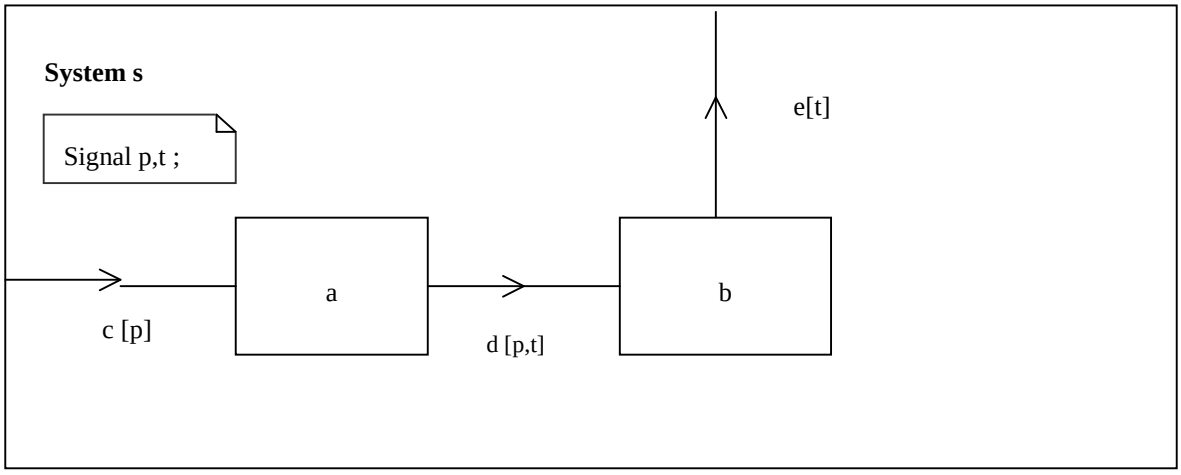


Figure 2. **Communication entre blocs.**

Les routes quant à elles permettent la communication entre deux processus et entre un processus et l'environnement externe au bloc auquel il appartient. Dans la figure 3, le bloc a est constitué des deux processus « a1 » et « a2 ». Le processus « a1 » peut envoyer les signaux « p » et « v » via la route « r2 » vers le processus « a2 » et peut recevoir le signal « p » de l'environnement via la route « r1 ». Le bloc « a2 » peut envoyer les signaux « p »et « t » vers l'environnement via la route r3.

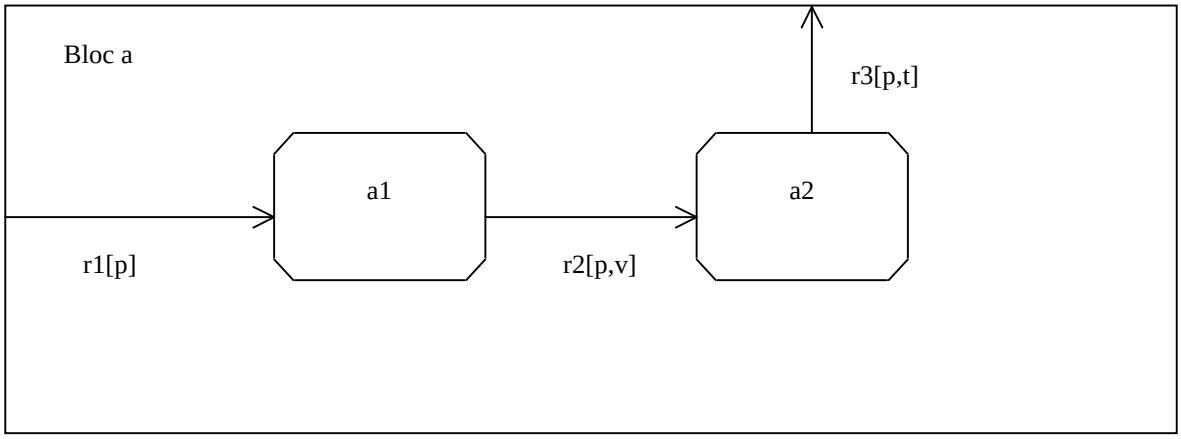


Figure 3. **Communication entre processus.**

Un signal est donc envoyé par un processus émetteur, transporté via des canaux et des routes, stocké dans la queue du processus récepteur et consommé par ce même processus. La figure 4 illustre la transmission d'un signal d'un processus vers un autre. A l'état e1, le processus « a1 » peut recevoir le signal « A ». Suite à cette réception, il effectue la tâche « T », puis envoie le signal « s » via « r2 » vers le processus « a2 » puis passe à l'état « e2 » . Ce dernier processus, étant à l'état e3 , reçoit le signal « s », effectue l'affectation du « id » ème élément du tableau « tab » à la variable « x », puis envoie le signal « s2 »via la route « r3 ».

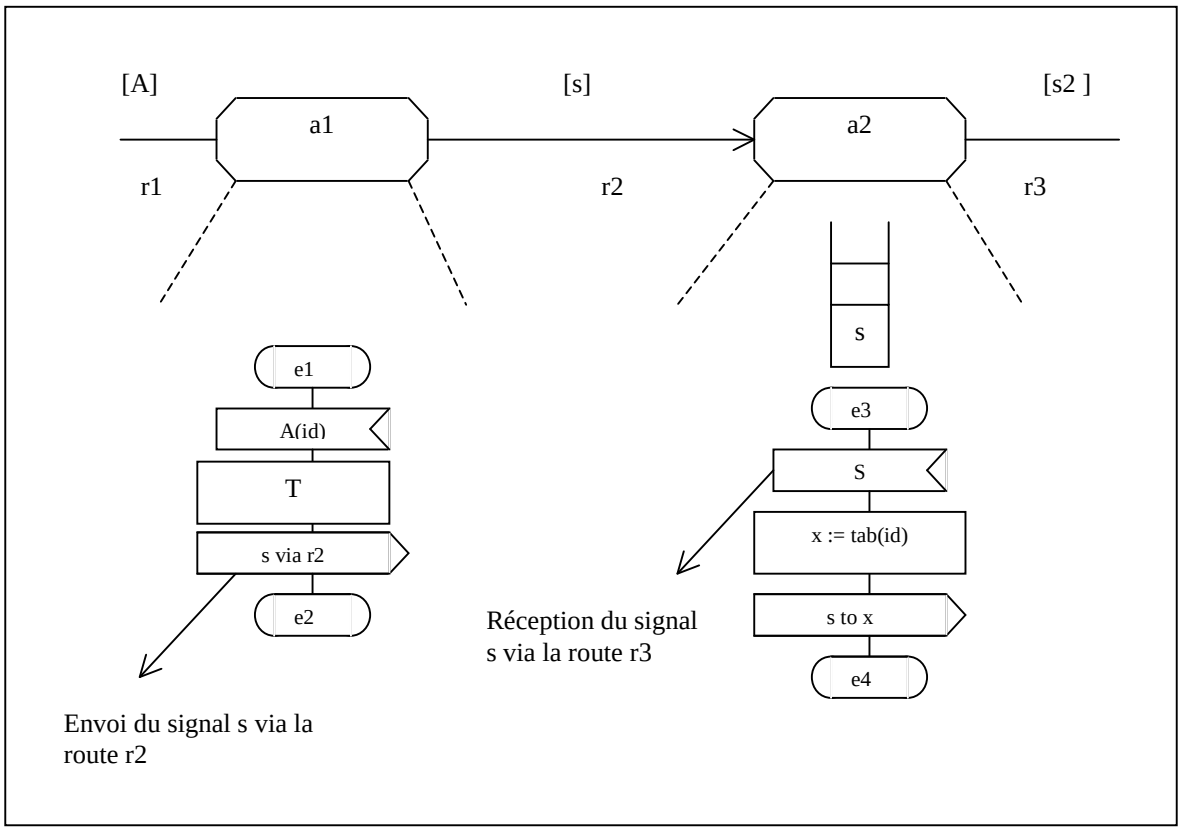


Figure 4 **. Etapes de communication entre deux processus en SDL.**

II.4. Aspect comportemental

La description du comportement d'un système en SDL est réalisé à travers la description de ses processus. Ces derniers sont décrits par des automates d'états finis constitués d'un ensemble d'états et de transitions.

Un processus est une entité pouvant être instanciée plusieurs fois (template). Chaque instance d'un processus est identifiée par une identité unique PId *(Process Identity)*. Elle dispose aussi de quatre informations de ce type :

- self : l'identité de l'instance elle même ;
- sender : l'instance émettrice d'un signal ;
- parent : l'instance créante ;
- offspring : l'instance récemment créée.

Une instance d'un processus possède aussi les caractéristiques suivantes :

- un temps de vie : une instance est créée soit à l'initialisation du système soit dynamiquement par une instance. Son existence cesse quand le symbole « stop » est rencontré (fig 5.c) ;
- des paramètres : valeurs affectées par le processus parent lors de la création dynamique de cette même instance ;
- des variables : qui, en combinaison avec l'état courant représentent l'état complet de l'instance ;
- un port d'entrée : qui reçoit les signaux et les garde jusqu'à leur consommation ;

- un ensemble d'entrée et de sortie qui constituent des signaux pouvant être reçus ou émis par ce même processus ;
- un nombre minimum et maximum d'instantiation qui doit être précisé dans la définition du processus sous forme d'un couple (nombre $_{minimal}$, nombre$_{maximal}$).

Un processus est représenté par des états (fig 5.a) qu'il peut prendre et des transitions entre ces états. L'interprétation d'un processus commence par l'affectation des paramètres formels, la création et l'initialisation des variables et l'interprétation du symbole « start » (fig 5.b). Cette interprétation s'arrête quand le symbole « stop » est rencontré (fig 5.c).

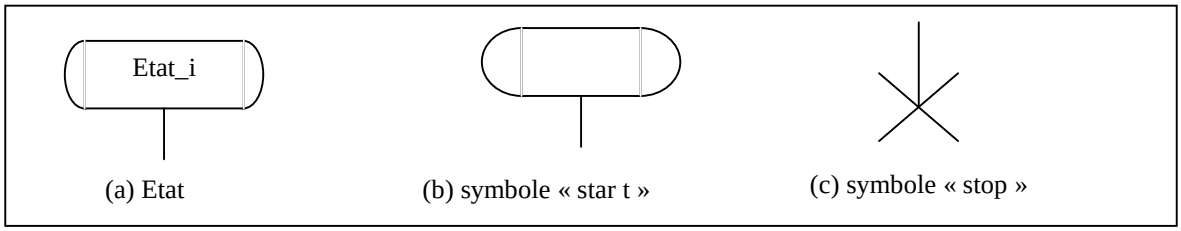


Figure 5. **symboles « start » et « stop ».**

Quand un processus est dans un état donné, Il peut accepter des stimulis via son port d'entrée. ces stimulis peuvent être des signaux ou des expirations de temporisateurs. Le même état peut apparaître à différents emplacements de la machine d'états du processus.

Le déclenchement d'une transition à partir d'un certain état d'un processus se fait par un signal d'entrée (fig 6.a). Il est aussi possible d'initier une transition par des signaux continus. Un signal continu est décrit sous forme de condition devant être satisfaite pour le déclenchement de la transition (fig 6.b). Un signal d'entrée peut indiquer aussi les valeurs des messages qu'il transportent et qui peuvent être affectées à des variables locales du processus.

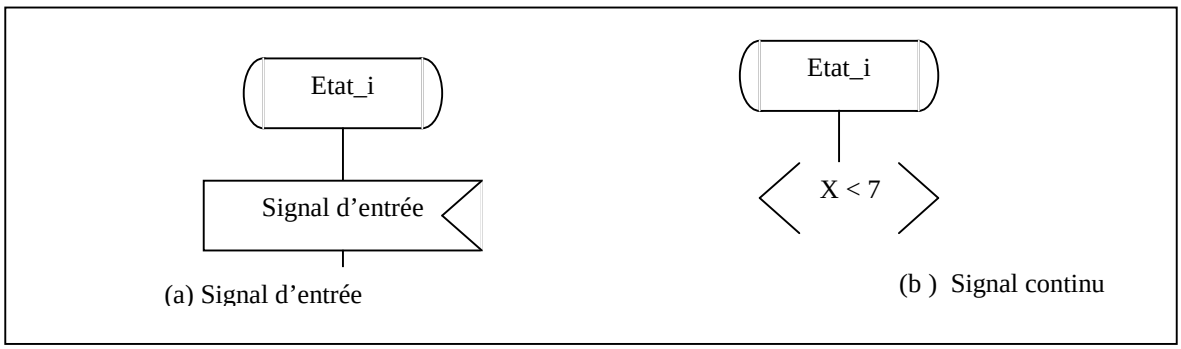


Figure 6. **Etat, signal d'entrée et signal continu.**

Une transition consiste en une ou plusieurs actions. Une action peut être soit un envoi de signal, une tâche de changement de valeur de variable, une création de processus, une décision, un appel de procédure, une initialisation ou réinitialisation de temporisateur.

L'envoi d'un signal est indiqué par le symbole de sortie (output) (fig 7.a). Dans ce type d'action, on peut aussi préciser des valeurs qui seront transportés sous forme de paramètres

du signal envoyé. Il y a aussi possibilité d'indiquer le processus récepteur du signal envoyé par le mot clé « to » ainsi que les routes et/ou canaux par le mot clé « via ».

Les changements des valeurs des variables sont représentés par le symbole de la figure 7.b. Cette action peut consister en des affectations, des initialisations ou réinitialisation de temporisateurs ou des opérations d'export/import de variables.

La création dynamique d'une instance de processus peut se faire en SDL par une autre instance d'un autre processus au sein du même bloc. Cette création est indiquée par le processus de la figure 7.c. Durant la création, les valeurs des paramètres effectifs sont affectées aux paramètres formelles.

Les choix entre deux alternatives du comportement d'un processus se fait par le symbole de décision (fig 7.d). Un choix consiste en une condition et deux branches de comportement.

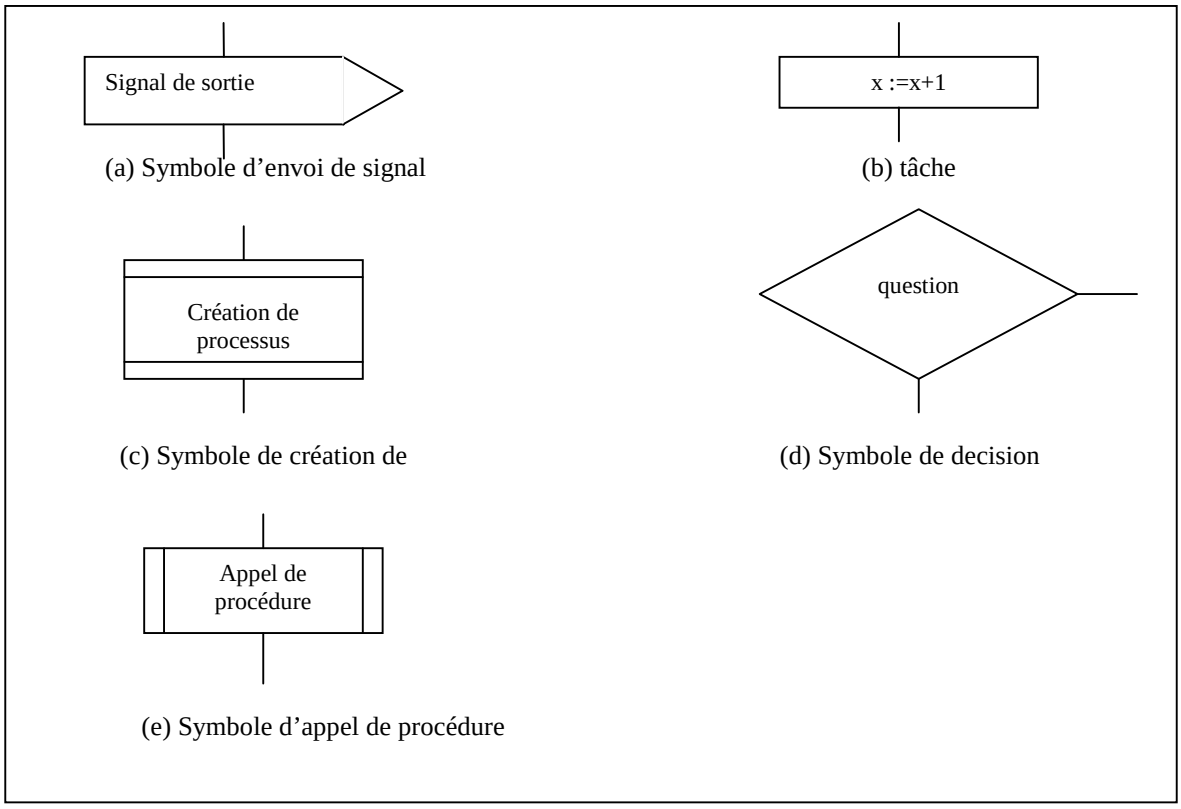


Figure 7. **Actions d'une transition.**

A des fins de réutilisation et de modularité au sein d'un processus, il y a possibilité de définir des procédures. L'appel de ces procédures est représenté par le symbole de la figure 7.e. Cet appel se fait éventuellement avec passage de paramètre effectifs. Une procédure représente une partie du comportement d'un processus et donc utilise les mêmes symboles présentés dans la figure 7, sauf que les symboles de début et de fin diffèrent de celle du processus et sont présentés dans la figure 8.

Pour la manipulation et la supervision des aspects temporels, le langage SDL a prévu la notion de temporisateurs. On peut ainsi lire la valeur du temps à un instant donné avec le

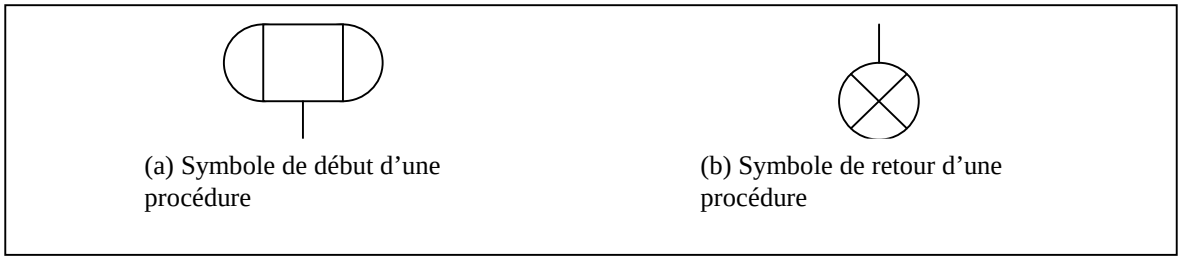


Figure 8. **Symbole de début et de retour d'une procédure.**

mot clé « now », réagir aux expirations des temporisateurs, les initialiser et réinitialiser (set et reset), inspecter leur statut (active ou non active).

II.5. Types abstraits de données

Les variables utilisées par un processus dans une spécification SDL sont typés. Les types peuvent être soit prédéfinis soit définis pas l'utilisateur. Les types de données en SDL sont définis par les « types abstraits de données » (TAD). Un type abstrait de données définit un ensemble de valeurs et une variable déclarée de ce type ne peut recevoir que ces valeurs.

Les types abstraits de données en SDL sont de trois catégories principales : les types simples, les types structurés et les types générateurs. Dans ces trois catégories, on distingue les types prédéfinis et les types définis par l'utilisateur. Le tableau 1 donne une idée sur les différents types prédéfinis.

Types prédéfinis	Nom des types
Types simples	Entier, réel, caractère, booléen, durée, temps et Pid
Types structurés	Chaîne de caractères
Types générateurs prédéfinis	Tableau et liste de PId

Tableau 1. **Types de données prédéfinis de SDL.**

L'objectif d'un type abstrait de données est la description d'un ensemble de valeurs, les moyens de les manipuler et les résultats de ces manipulations. Un nouveau type est défini par le mot clé « newtype ». La définition d'un type en SDL est divisée en trois parties : la description d'un ensemble de litéraux (constantes), la définition des opérations qui peuvent être effectuées sur les valeurs du type et la description des résultats produits par ces opérateurs en utilisant des axiomes. La figure 9 donne un exemple de déclaration d'un type couleur, de la déclaration d'une variable de ce type et d'un usage de cette même variable. Ce type définit trois valeurs constantes (blanc, rouge et rose), une opération de mixage de deux couleurs et un axiome définissant le résultat de cette opération sur les couleurs bleue et rouge.

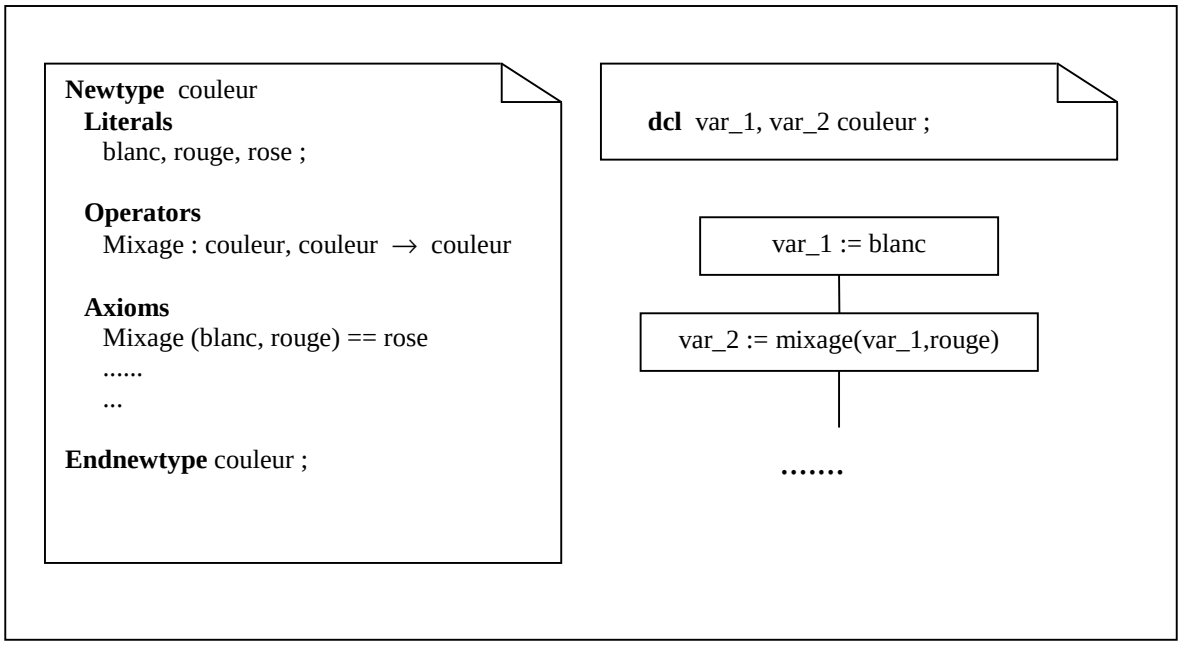


***Figure 9. Exemple d'un* type de données abstrait défini par l'utilisateur.**

En SDL, on peut utiliser aussi la notion de générateur. Ce concept permet d'avoir des types paramétrés. Il est utilisé pour décrire un type générique qui, en l'instanciant donne la définition d'un nouveau type. L'exemple de la figure 10 montre un type générique d'une pile paramétré par les éléments de la pile (TYPE t), le nom de l'opérateur indiquant sa longueur (OPERATOR Length), le nom du litéral indiquant une pile vide (LITERAL Empty), une valeur de la longueur maximale (CONSTANT maxlength).

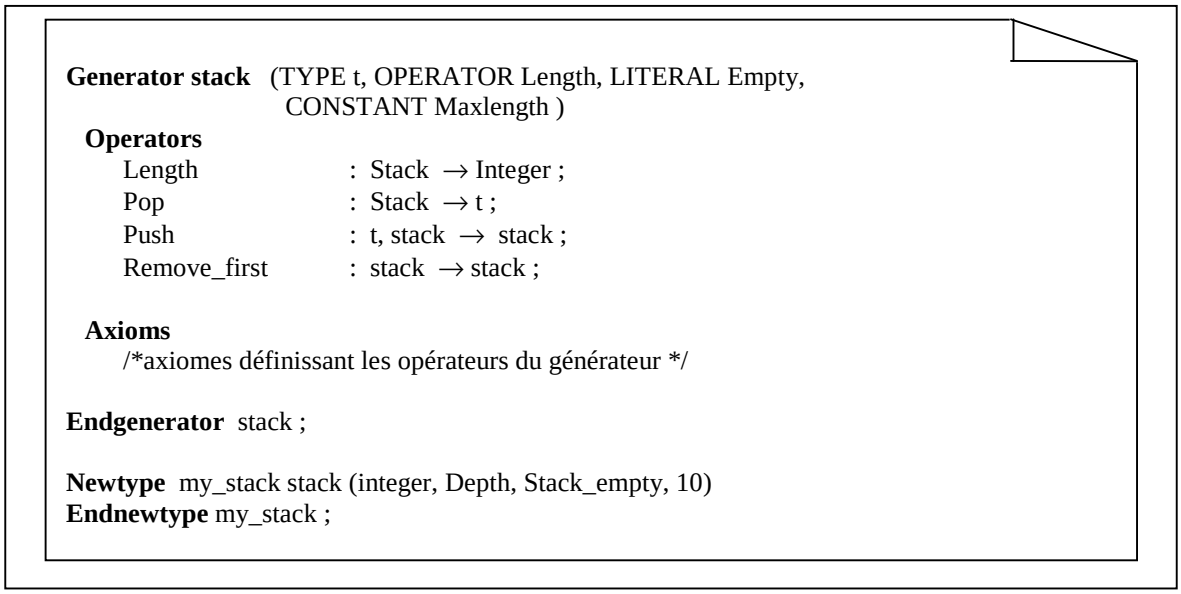


***Figure 10. Exemple d'un* type abstrait de données défini par l'utilisateur.**

La figure 10 montre aussi une utilisation du générateur pile. Ainsi, le type « my_stack »est défini par le type entier des éléments de la pile instanciée, l'opérateur profondeur (« depth »), le litéral Stack_empty et un maximum d'éléments de la pile de dix.

Nous avons essayé de présenter dans ce paragraphe quelques concepts principaux relatifs à l'aspect données. Pour plus d'informations sur les types abstraits de données, nous recommandons la norme Z100 [ITU-T 93].

II.6. Aspects objets et non déterminisme dans SDL

II.6.1. Demande de participation

L'avènement des concepts de l'orientés-objets comme nouvelles techniques adoptées par les méthodes de conception et les langages de programmation, a été d'un très grand apport pour le développement des systèmes et applications distribués. C'est dans ce sens que les concepteurs du langage SDL lui ont apporté des extensions tout en gardant les concepts du SDL'88. Le nouveau langage, portant le nom de SDL'92, permet alors de supporter de tels concepts (orientés-objets). Ainsi, dans SDL'92, les types « système », «bloc », « processus », et « service » peuvent être instanciables. Ces types disposent de ports qui peuvent être connectés aux entités de communication (canaux et routes). SDL'92 considère aussi, les signaux, procédures et types comme entités instantiables [Turn 93].

Un type général (*supertype*) peut être spécialisé en plusieurs types spécifiques (*subtype*). Dans cette spécialisation, le type spécifique hérite les propriétés du type général. Une spécialisation permet à un type spécifique :

- d'avoir de nouvelles propriétés ajoutées à celles du type général ;
- et la redéfinition des propriétés abstraites du type général.

Les propriétés peuvent être, par exemple des définitions incluses dans le type, dans les transitions pour les types processus, dans les types service et procédures.

Les types peuvent être paramétrisés. Cette paramétrisation a pour avantage la réutilisation des définitions d'un même système dans différents contextes. Le concept « package », ensemble de types, permet quant à lui, la réutilisation à l'intérieur des systèmes.

II.6.2. Non-déterminisme

SDL'92 dispose de deux constructeurs pour la description du non-déterminisme : les transitions spontanées et les valeurs indécidables.

Une transition spontanée permet l'initiation non-déterministe des transitions. Plusieurs transitions spontanées peuvent être relatives à un même état. Une transition spontanée est dénotée par une extension du format pour une entrée ayant la valeur none.

Une valeur indécidable est générée par le nouveau opérateur. Cet opérateur prend comme opérande un type et retourne une valeur de ce même type. Une expression any possède le format suivant : ***any*** (*identificateur_type*). Comme exemple, ***any*** (*boolean*) retourne *True* or *False* de manière non-déterministe.

III. Spécification informelle d'un système de conférence à distance

Afin de pouvoir décrire de manière informelle le système de conférence à distance, nous allons essayer de décrire quelques aspects d'une conférence réelle. Généralement, un colloque scientifique - et en particulier informatique – se déroule en plusieurs journées (minimum trois). La première journée est consacrée à des tutoriels. Ces tutoriels se déroulent sous forme d'exposés semblables à ceux d'un séminaire. Dans ce type de présentation, il peut y avoir plusieurs interactions entre l'assistance et le conférencier et la coordination (question/réponse) est gérée par ce dernier lui même. Cet aspect du colloque peut être assimilé à des cours et peut être très bien géré par un système de télé-enseignement [Owez 97].

Les autres journées d'un colloque sont organisée sous forme de sessions. Plusieurs sessions peuvent avoir lieu en même temps. Une session est initiée par un participant du colloque qui devient dès lors le président (chairman) de cette session. Le chairman aura la tâche de gestion de la session. Cette gestion consiste en une coordination d'acquisition de la parole entre les différents acteurs de la session y compris le chairman lui même. C'est ce type de présentation auquel on va s'intéresser le plus dans notre spécification.

En général, une session se déroule selon le scénario décrit dans ce qui suit. Lorsqu'un chairman initie une session, il commence par inviter le premier conférencier. Celui-ci peut être présent ou absent. Dans le premier cas, le chairman lui donne la parole en lui accordant quinze ou vingt minutes pour présenter son travail. Le chairman peut réagir si le conférencier dépasse le temps qui lui a été accordé. Après avoir terminé l'exposé, ce dernier rend la parole au chairman. Celui-ci entame la phase de discussion et de questions posées par l'assistance. Le chairman donne ainsi la main à des participants à tour de rôle pour poser des questions auxquelles le conférencier doit répondre. Le chairman peut aussi arrêter cette phase pour des raisons de contraintes temporelles. Ce scénario sera répété jusqu'à ce que prenne fin la session. Toutes ces interactions relèvent des aspects coordination au sein d'une session. Notre objectif est de spécifier formellement ces aspects en prenant en compte les contraintes temporelles.

Maintenant qu'on a précisé quelques aspects d'un colloque, nous allons décrire les besoins requis du système à spécifier pour supporter de tels aspects. Nous supposons qu'une annonce ainsi qu'un programme du colloque sont diffusés auparavant. Le système doit présenter un ensemble de services aux différents participants du colloque. Ainsi, il doit permettre à une personne de joindre la conférence. Une fois présent sur la liste des participants, une personne doit pouvoir entrer dans une session et/ou la quitter. Elle doit aussi pouvoir initier une session si elle est autorisée. Notre système doit pouvoir assurer la coordination entre les différents acteurs d'une même session. Pour se faire, on a opté pour la solution de jeton [Fric 96]. Ainsi, lorsqu'un chairman invite un participant et que ce dernier répond à cette invitation (présence), le système transfert automatiquement le jeton du chairman vers le participant. Ce transfert s'accompagne par la diffusion de ce dernier ainsi que ces transparents vers les autres stations des participants de la même session. Cette tâche est effectuée par le système de visioconférence. Lorsque le conférencier termine son exposé, il rend le jeton au chairman. Ceci implique automatiquement la rediffusion du chairman sur les stations des autres participants de la session. Si le participant dépasse son temps d'exposé, le système peut lui retirer le jeton et le donner au chairman auquel cas sa diffusion par le système de visioconférence s'arrête. Lors de la

phase des questions, le jeton va circuler entre le chairman, l'assistance et le conférencier. Là aussi, la diffusion se fait en fonction du jeton.

La figure 11 donne l'architecture générale de notre système. Dans cette architecture, on a représenté les participants réels sous forme de petits cercles. Ces participants forment des groupes qui représentent les sessions de la conférence (représenté par un grand cercle). Le chairman est représenté par un cercle gris. A chaque participant correspond un processus au niveau du système de coordination (représentés par des carrés).

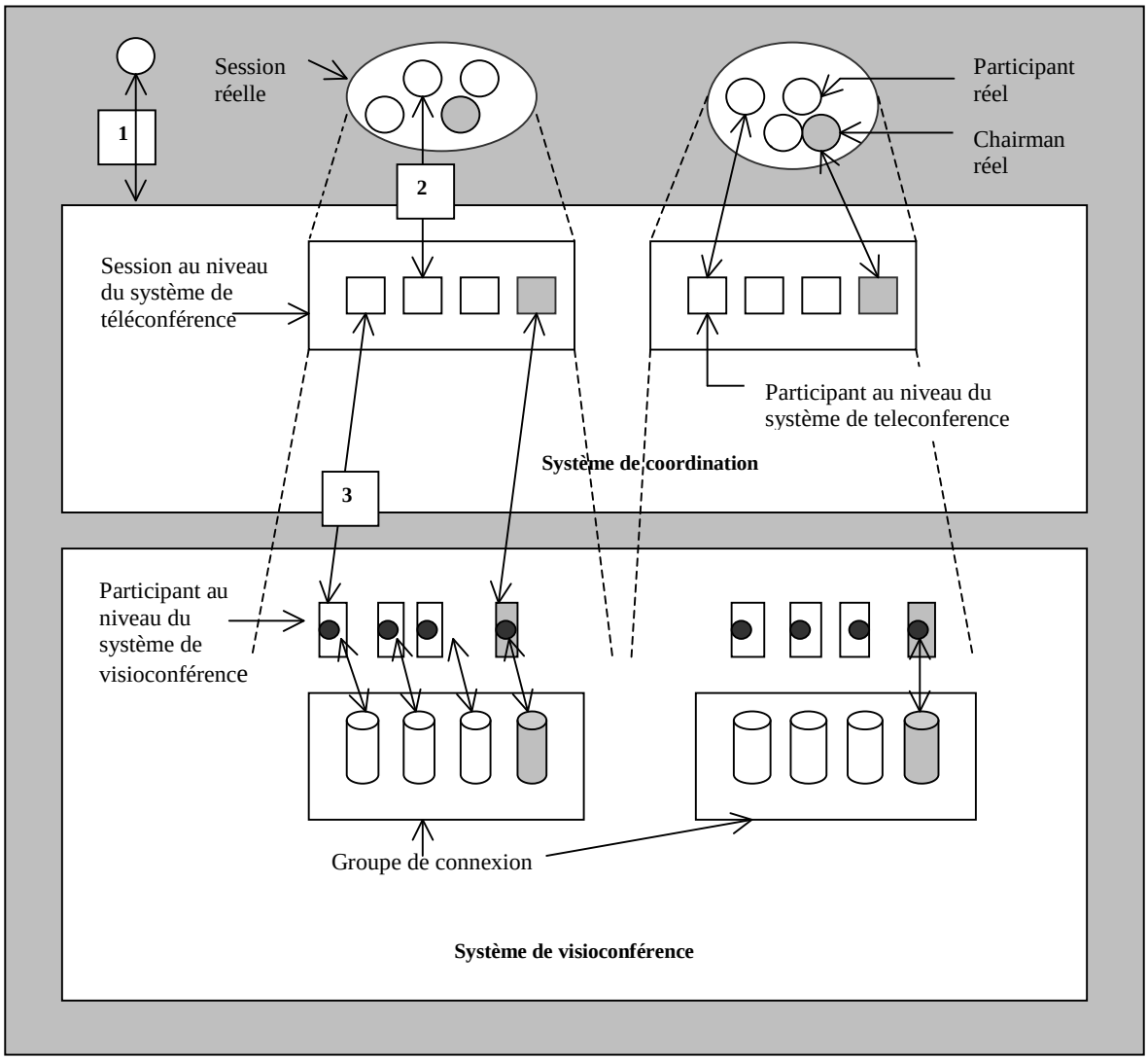


Figure 11. **Architecture générale du système de téléconférence.**

Un participant réel peut interagir avec le système pour participer à la conférence (interaction numéro dans la figure). Une fois son processus créé, il peut initier une session ou entrer dans une session via des services présentés par son processus correspondant (interaction numéro 2). Ces processus sont regroupés de manière à former une session au niveau du système de coordination.
Chacun de ces processus dispose d'un processus (représentés par des carrés avec un petit cercle noir) au niveau du système de visioconférence ainsi que d'une connexion. Les connexions sont regroupées de manière à former une session au niveau du système de

visioconférence. Un processus au niveau système de coordination peut demander des services de diffusion ou d'arrêt de diffusion au processus du niveau visioconférence correspondant. Ces demandes se font en fonction d'acquisition ou de perte de jeton. Suite à ces demandes, le processus du niveau visioconférence demande au groupe de connexion de diffuser ou d'arrêter la diffusion de la personne correspondante.

IV. Spécification formelle d'un système de téléconférence en SDL.

Dans cette partie, nous allons procéder à la description de la spécification formelle du système de téléconférence décrit au paragraphe précédent. Cette description comporte la présentation de l'architecture du système SDL sous forme de blocs et de canaux, la description de chaque bloc en présentant ses différents processus et enfin, la description de quelques processus.

IV.1. Architecture globale de la spécification

En respectant le schéma présenté au paragraphe III, l'architecture du système SDL sera décomposé en deux blocs principaux : le bloc « coordination » et le bloc « visioconférence ». Le premier bloc gère l'aspect coordination au sein des différentes sessions, l'accès d'un participant à la conférence, son accès à une session. Il présente différents services à l'environnement externe (fig 12). Ces services sont représentés par des signaux que le système peut accepter de l'environnement externe. Ces signaux sont décrit dans le tableau 2.

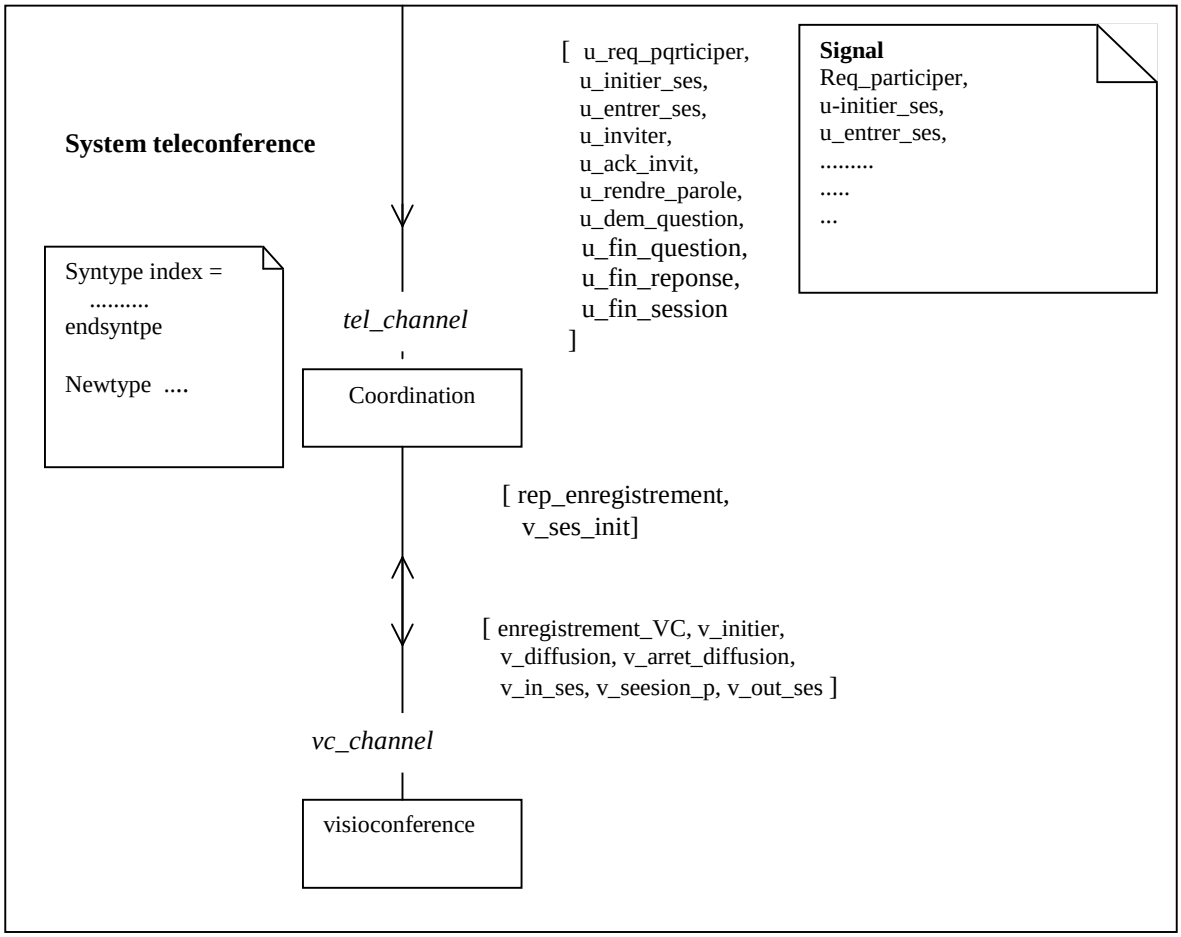


Figure 12. **Architecture SDL du système de téléconférence.**

Service	Description du service
u_req_participer	Service de demande de participation.
u_initier_ses	Service de demande d'initiation d'une session.
u_entrer_ses	Service de demande d'entrée dans une session.
u_inviter	Service de demande d'invitation d'un participant par un chairman.
u_ack_inviter	Service de réponse positive à une demande d'invitation.
u_rendre_parole	Service de demande de rendre la parole par un participant après son exposé.
u_dem_question	Service de demande de question au conférencier.
u_fin_question	Service d'indication de la fin de question.
u_fin_reponse	Service d'indication de la fin de réponse.
u_fin_session	Service de d'indication de fin de session.

Tableau 2. **Services présentés aux utilisateurs.**

Le bloc « visioconférence » s'occupe des aspects de transmission d'audio/vidéo des différents participants ainsi que des documents qu'ils présentent. Il met à disposition du bloc « coordination » un ensemble de services. Ces services sont présentés au tableau 3. Parmi ces services, on trouve ceux de diffusion et d'arrêt de diffusion. Ces services se font selon la possession du jeton d'un processus participant au niveau du bloc « coordination ».

Service	Description du service
enregistrement_VC	Service de demande d'enregistrement d'un nouveau participant au niveau du système de visioconférence.
v_initier	Service de demande de création d'une session au niveau du système de visioconférence (futur groupe de connexion).
v_diffusion	Service de demande de diffusion invoqué par un participant au niveau du bloc « coordination » sur son processus participant correspondant au niveau du bloc « visioconférence ».
v_arret_diffusion	Service de demande d'arrêt de diffusion invoqué par un participant au niveau du bloc « coordination » sur son processus participant correspondant au niveau du bloc « visioconférence ».
v_in_ses	Service de demande d'ajout d'une connexion correspondant à un participant du bloc « coordination » à une session au niveau d'une session du bloc visioconférence.
v_out_ses	Service de demande de suppression d'une connexion correspondant à un participant du bloc « coordination » à une session au niveau d'une session du bloc visioconférence.

Tableau 3. **Services présentés par le bloc de visioconférence au bloc coordination.**

Dans la suite de cette description nous allons nous contenter de la description du bloc « coordination » et quelques processus de ce bloc. La spécification complète tel qu'elle a été réalisée avec l'outil ObjectGeode est présenté en annexe de ce rapport.

IV.2. Description du bloc « coordination »

Rappelons que le bloc « coordination » a pour rôle de gérer les différentes interactions qu'on peut avoir entre participants dans une conférence et en particulier au sein d'une session. Ce bloc est constitué de trois principaux processus : le processus « monitor », le

processus « participant » et le processus « session » (fig 13). Le processus « monitor » interagit avec l'environnement externe via les routes « monitor_route1 » et « monitor_route_2 ». La première route est liée au canal « tel_chanel » et la deuxième au canal « vc_channel ». L'instance du processus « monitor » peut créer des instances des processus « participant » et « session ». Elle peut par la suite communiquer avec ces dernières respectivement via les routes « monitor_participant_route » et « monitor_session_route ». Après sa création, une instance du processus participant peut interagir avec l'utilisateur, pour lequel elle a été créée, via la route « participant_route1 » pour lui assurer quelques services. Elle peut aussi interagir avec le bloc « visioconférence » via la route « participant_route2 » pour lui demander des services.

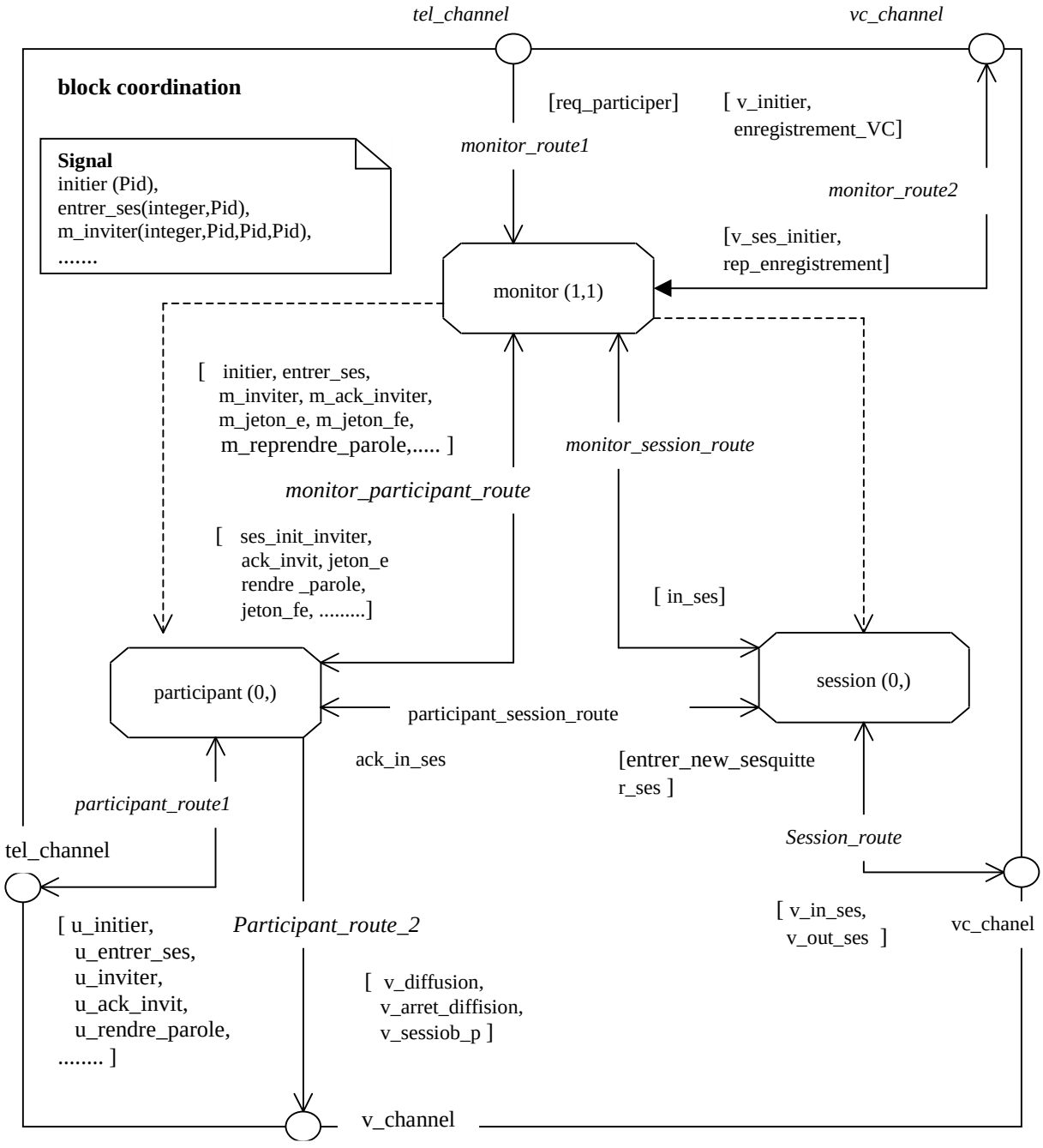


Figure 13. **Bloc « coordination »**

Une instance du processus « session » peut interagir avec le bloc « visioconférence » via la route session_route. Elle peut aussi communiquer avec une instance du processus « participant » via la route « participant_session_route ». Dans le paragraphe suivant, nous allons présenter la description de quelques parties des processus pour expliquer l'échange de quelques signaux et l'interaction de quelques processus.

IV.3. Description de l'interaction entre processus du bloc « coordination »

Dans ce paragraphe nous présentons quelques interactions entre les processus du bloc « coordination ». Nous allons commencer par présenter le service d'accès à la conférence par un utilisateur du système puis nous allons détailler une partie des échanges ayant lieu entre les processus du bloc «coordination » en vue de la gestion d'acquisition de parole entre les différents acteurs d'une session.

IV.3.1. Demande de participation

Lorsque un utilisateur désire joindre la conférence, il procède par l'envoi du signal « req_participer » (fig 14). Ce signal sera acheminé via le canal « tel_channel » puis via la route « monitor_route » pour être inséré dans la file d'attente du processus « moniteur » à l'état « svce » (pour service). Lorsque ce processus reçoit ce signal, il demande au bloc « visioconférence » d'enregistrer le nouveau participant par l'envoi d'un signal « enregistrement_VC » au processus « SVM » (système de visioconférence) du bloc « visioconférence ». Cet enregistrement se fait par la création d'un processus « v_participant » et d'une connexion relative au nouveau participant au niveau du bloc « visioconférence ».

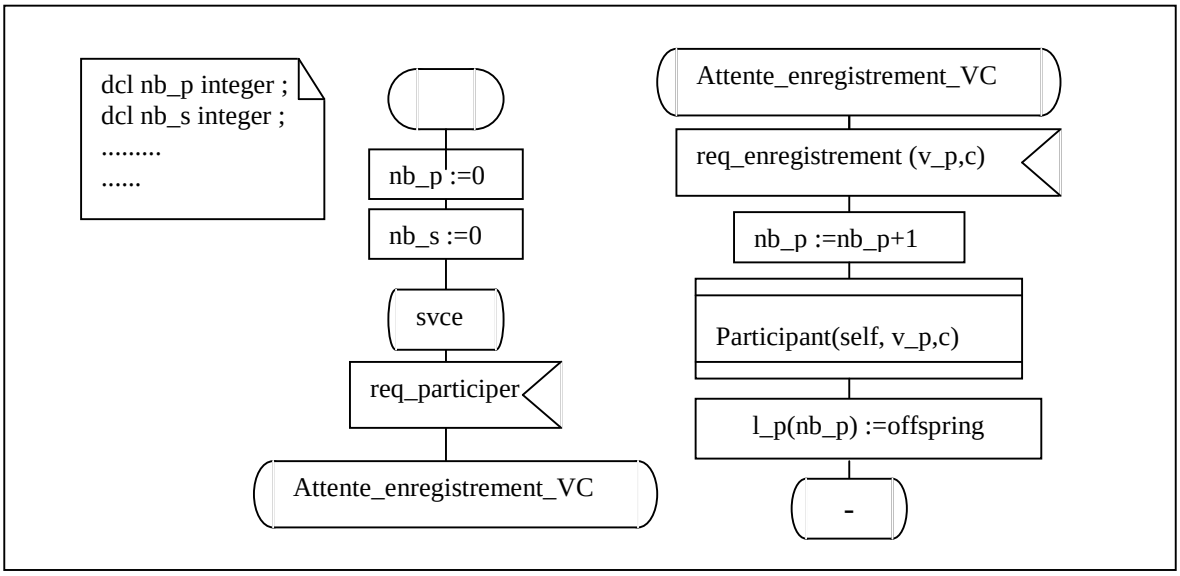


Figure 14. **Demande de participation.**

Le processus « v_participant » (v :pour visioconférence), assure les services de diffusion et d'arrêt de diffusion. Une fois que le processus « monitor » reçoit la réponse d'enregistrement (« rep_enregistrement »), il crée un nouveau processus « participant » avec lequel va interagir le nouveau utilisateur durant toute sa participation à la conférence. Ce processus va offrir un certain nombre de services à l'utilisateur externe correspondant.

IV.3.2. Interactions de coordination

Dans ce paragraphe, nous décrirons la spécification des interactions relevant de la coordination de l'acquisition de parole au sein d'une session de la conférence. Nous supposons qu'une session ou plusieurs sessions sont créées et qu'un ensemble de participants ont joint la conférence. Dans une session, le chairman commence par inviter le premier conférencier. Pour cela, il envoie le signal « u_inviter » (le « u » pour indiquer que c'est un utilisateur externe) à son processus « participant » correspondant en précisant le numéro du participant « pp » (branche 2, figure 15). Ce dernier, une fois le signal « u_inviter » reçu, envoie le signal « m_inviter » au processus « monitor » qui, en recevant ce même signal, envoie le signal « inviter » au processus « participant » ayant pour indice « pp » dans la liste des participants et initialise le temporisateur THA (temps d'appel). Ce processus passe ensuite à l'état « attente_rep_invit » (attente de réponse d'invitation).

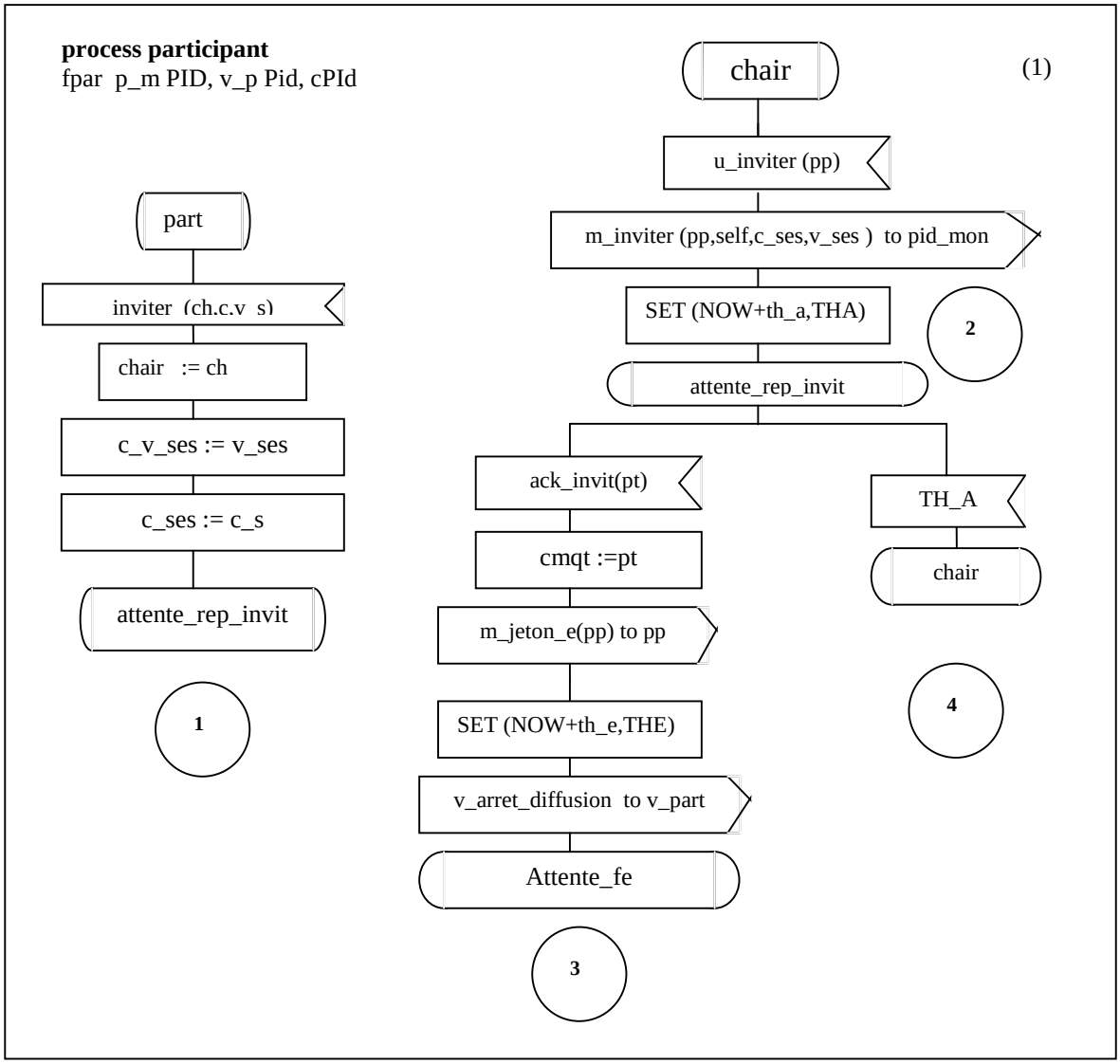


Figure 15. **Interaction d'invitation d'un participant par un chairman**

Lorsque le processus « monitor » reçoit le signal « m_inviter », il envoie le signal « inviter » au processus « participant » relatif au conférencier invité (qu'on va designer par processus conférencier), puis passe à l'état «attente_ack_invit » (branche 1 de la figure 16).

En recevant ce signal, le processus conférencier passe à l'état « attente_rep_invit ». Si la personne invitée répond à l'invitation par envoi du signal « u_ack_invit » (réponse à l'invitation), son processus « participant » effectue une série de vérifications pour déterminer si le conférencier en question participe déjà à une autre session, et le cas échéant si la session proposée par le chairman n'est pas la même que celle à laquelle le conférencier assiste. Suite à ces vérifications, il envoie le signal « m_ack_invit » au processus « monitor » (branche 1 figure 17) et passe à l'état « attente_parole_e » (le e pour exposé). Celui-ci, en recevant ce signal, envoie le signal « ack_invit » au processus relatif au chairman (branche 2 figure 16) qui envoie le signal « m_jeton_e » au processus « monitor » et initialise le temporisateur THE (temporisateur d'exposé) puis passe à l'état « attente_parole_fe » (attente de fin d'exposé). A la réception du signal « m_jeton_e », le processus « monitor » transmet le signal « jeton_e » au processus conférencier (branche 4 figure 16).

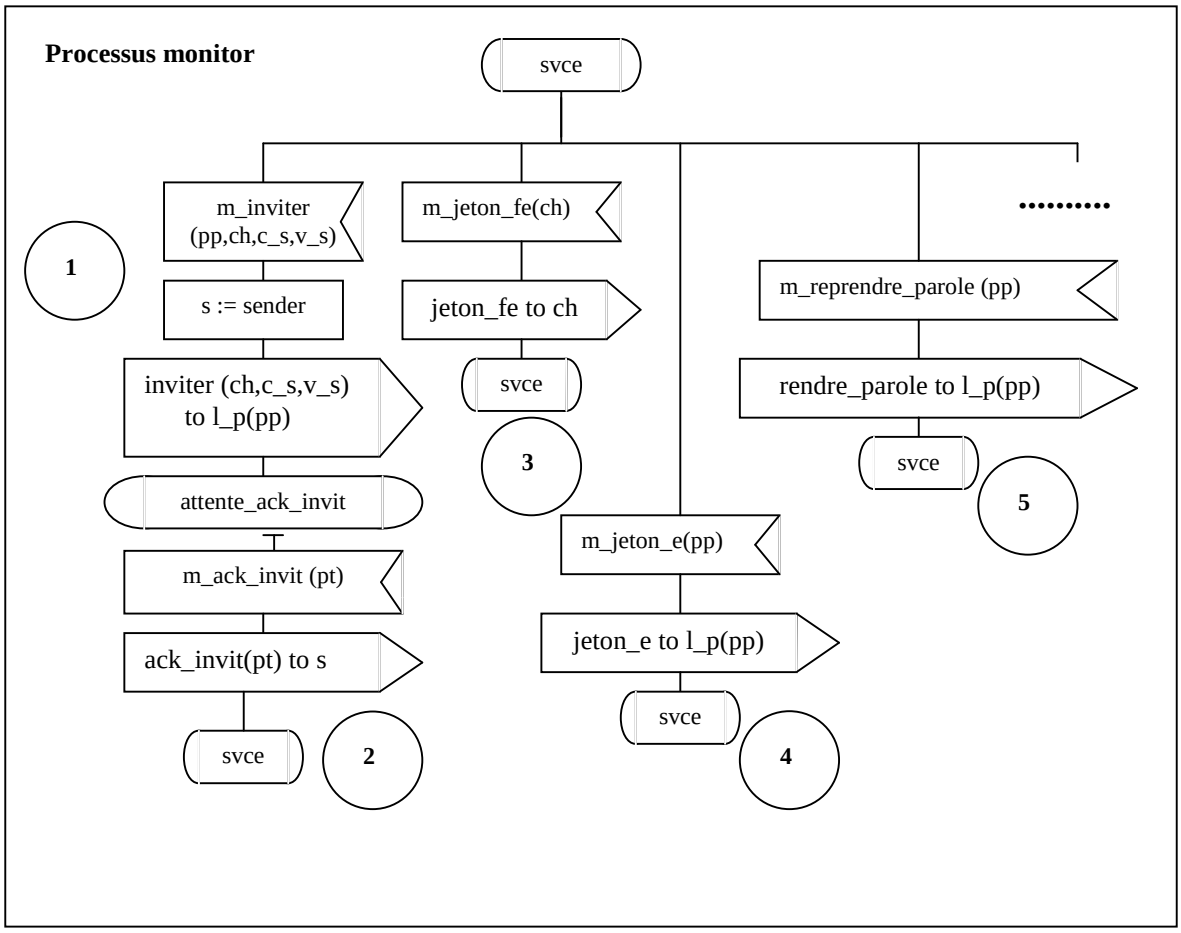


Figure 16. **Processus Monitor.**

Lorsque ce dernier reçoit ce signal, il demande à son processus « v_participant » de lancer la diffusion du conférencier ainsi que de ses transparents (branche 2 figure 17). Il passe ensuite à l'état «exposé ». Si le conférencier termine son exposé avant l'expiration du temps accordé, il envoie le signal « u_rendre_parole » à son processus « participant ». Celui-ci va par la suite demander l'arrêt de diffusion en envoyant le signal « v_arret_diffusion » au processus « v_participant » (branche 3 figure 17), puis remet le jeton en envoyant le signal « m_jeton_fe » au processus « monitor ». Celui-ci remet le jeton au processus du chairman en envoyant le signal « jeton_fe » (branche 3 figure 16).

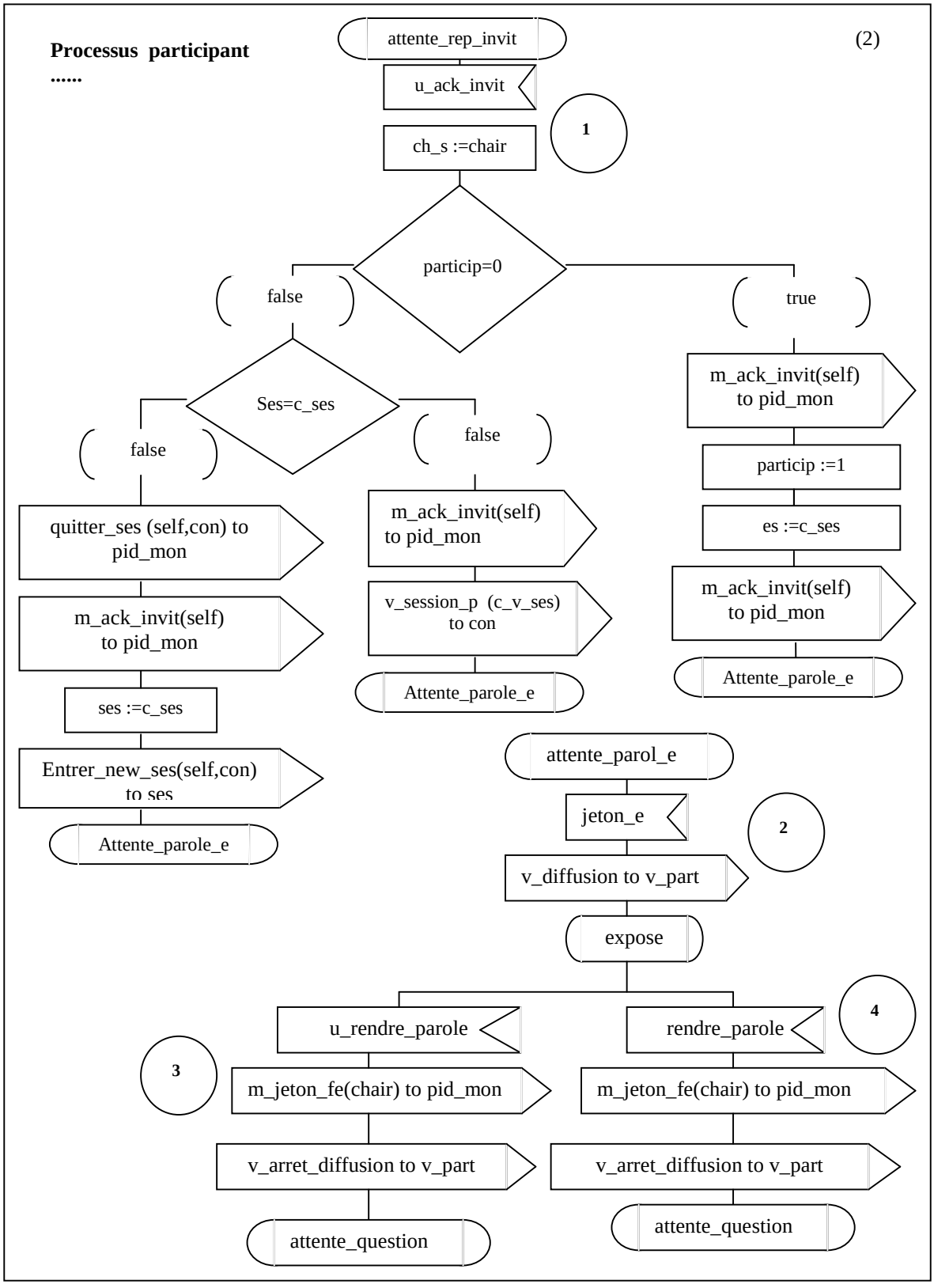


Figure17. **Vérifications des sessions (1) et réception du jeton par un participant (2,3et 4).**

Une fois ce signal reçu, le processus « participant » relatif au chairman relance la diffusion de ce dernier en envoyant le signal « v_diffusion » à son processus « v_participant » et

passe à l'état « phase_question » (branche 2 figure 18). Une autre éventualité peut se produire si le conférencier ne respecte pas le temps accordé à son exposé. Dans ce cas le signal d'expiration du temporisateur « THE » (branche 2 figure 18) va être reçu par le processus du chairman. Suite à cette réception, une suite d'interactions va être effectuée pour le retrait du jeton au processus du conférencier et par suite l'arrêt de sa diffusion et la rediffusion du chairman. Ce dernier passe par suite à l'état « phase_question ».

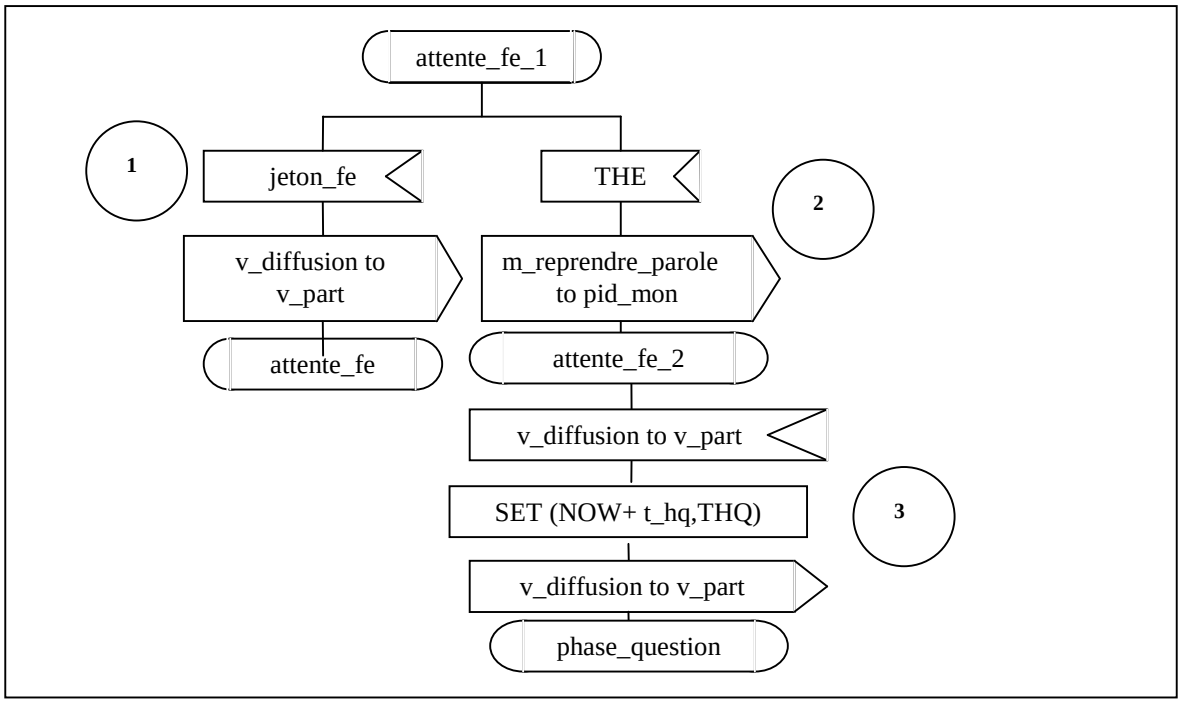


Figure18. **Récupération du jeton par un chairman.**

Pour la phase de question c'est la même stratégie qui a été suivie (jeton) pour gérer la coordination. Le jeton va être échangé entre le chairman, le conférencier et l'assistance de manière a ce que des participants à la session peuvent poser des questions, le conférencier peut leur répondre tout en maintenant une organisation des interventions (voir la suite du processus « participant » en Annexe). La coordination entre intervenant dans la phase question se déroule de façon à ce que c'est la personne disposant de la parole (jeton) qui sera diffusé. Ceci se fait par des appels aux services des processus « v_participant » du bloc « visioconférence ».

V. ObjectGeode

ObjectGeode est un ensemble d'outils qui a été conçu spécifiquement pour supporter le processus de développement d'application et systèmes distribuées, temps réels et complexes. ObjectGeode fournit des solutions pour :

- manipuler la complexité par la modélisation des différents aspects d'un système (architecture, données, comportement) avec une représentation graphique ;
- utiliser les standards internationaux appropriés tel que MSC (*Message Sequence Chart*), SDL et TTCN (*Tree and Tabular Combined Notation*) de l'UIT-T, et OMT (*Object Modeling Technique*) pour supporter les caractéristiques des systèmes temps réels. Les MSCs permettent d'avoir la description de scénario de haut niveau et des interactions des différents composants du système. SDL permet la conception

graphique des systèmes réactifs par des machines d'états finis communicants et OMT permet la description orienté-objet des besoins ainsi que des données ;

- produire des systèmes de qualité très élevé en appliquant des techniques de vérification de validation ;
- délivrer le système dans un temps record par l'automatisation et la génération d'applications fiables ;
- maintenir les applications à un niveau de conception graphique.

La figure 19 donne une idée sur les différents outils de ObjectGeode.

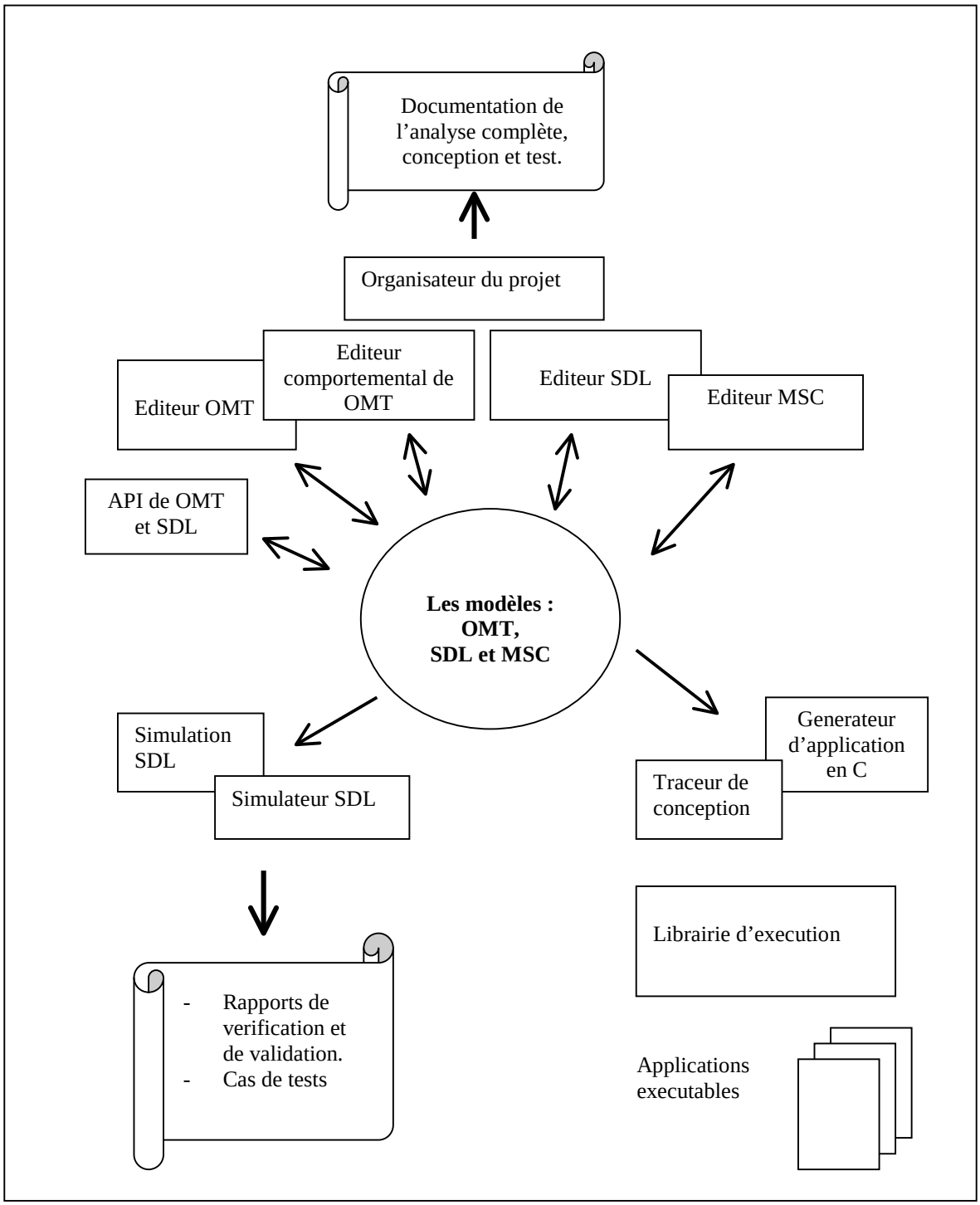


Figure 19. Outils de ObjectGeode

VI. Scénario de simulation

Dans ce paragraphe, nous allons procéder à la description d'un scénario de simulation de cette spécification. La figure 20 donne une description graphique de ce scénario. Les participants sont représentés par des cercles et les sessions par des carrés. Les différentes interactions sont représentées par des flèches avec les noms des interactions et un numéro d'ordre entre parenthèse. Les sessions peuvent se dérouler en parallèle. Les interactions parallèles seront représentées par le même numéro mais différenciée par le symbole «' ». On suppose aussi que tous les participants de cette figure sont tous passés par le service de demande de participation.

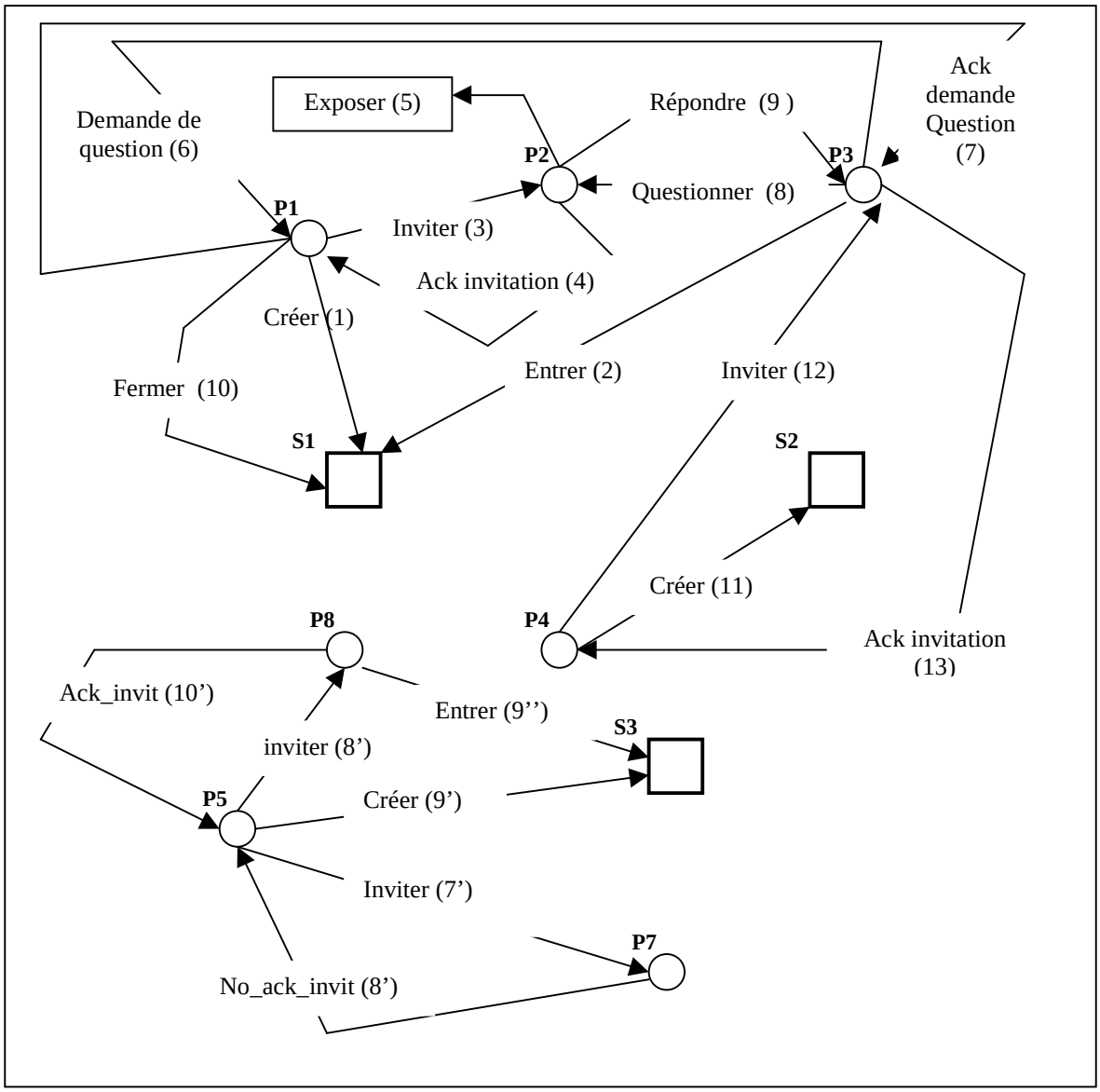


Figure 20. **Scénario de simulation de sessions d'une conférence à distance**

Ainsi, dans ce scénario, le participant P1 commence par créer une session, puis invite le participant P2, ce dernier répond à son invitation, aura la parole puis commencera son exposée. On supposera que ce dernier respectera le temps accordé. Après avoir terminé, il

rend la parole au chairman. Celuici initie dès lors la phase des questions et donne la parole au participant P3. Après, la parole sera donnée au conférencier (P2) pour donner sa réponse puis rendue au chairman P1. Ce dernier donne ensuite fin à cette session. En parallèle à la session S1, se déroule la session S2 dans laquelle on a considéré le cas où le participant P7 ne répond pas à l'invitation et le conférencier P8 ne respecte pas le temps accordé à une session. Dans la session S2, on a considéré le cas où le participant P3 sera invité alors qu'il participe déjà à une autre session.
Nous estimons que ce scénario supporte tous les cas possibles. Nous l'avons ainsi simulé, généré le fichier scénario, puis le fichier MCS à partir de ce dernier. Ce fichier nous a permis les différentes interactions de cette simulation. Nous avons ainsi parcouru toutes les branches de ce scénario. Ceci nous a été indiqué par le pourcentage de chaque processus, donné lors de cette simulation.

VII. Conclusion

Nous avons présenté dans ce rapport une spécification informelle d'une application de conférence à distance. Nous avons procédé à sa spécification formelle en utilisant le langage SDL'92. Puis nous l'avons réalisé avec l'outil ObjectGeode. Ce même outil nous a permis d'avoir une spécification graphique compréhensible. Il nous a aussi permis de faire des simulations de différents scénarios. Ainsi, par cette spécification, nous avons pu spécifier formellement les aspects liés à la coordination au sein d'une conférence et en particulier ses sessions. Nous avons aussi démontré que le langage SDL a priori destinée aux spécifications des systèmes de commutation et communication, peut très bien être utilisée pour spécifier de tels aspects. Toutes fois, il nous reste à effectuer des vérifications en vue de faire ressortir s'il y a de blocage ou autre défaut. Nous envisageons aussi de reprendre cette même spécification en appliquant une méthodologie de conception orienté objet ainsi que son langage correspondant Mondel [Boch 90].

Références

[Boch 90] G.. Bochmann, M.Barbeau, M.Erradi, L.Lecompte, P.Mondain-Monval and N.Williams, « *Mondel : An Object-Oriented Spécification Langage* ». Publication 748. Departement d'informatique et de recherche operationnelle, Universitee de Montreal. Novembre 1990.

[Bolo 98] J.Bolot, T.Turletti, « *Adaptative Error Control for Packet Video in the Internet* ». Proc.ICIP'96, Lausane, Switzerland, Step. 16-19, 1996.

[Fric 97] O.Frick, « Formal Description and Interpretation of Coordination Protocols for Teamwork ». International Workshop Trends in Distributed Systems (TREDS '96), Aachen, Germany,October 1996.

[ITU-T 97] International Telecommunication Union, « *Langage de description et de spécification du CITT »*. IUT-T Z100, 1993.

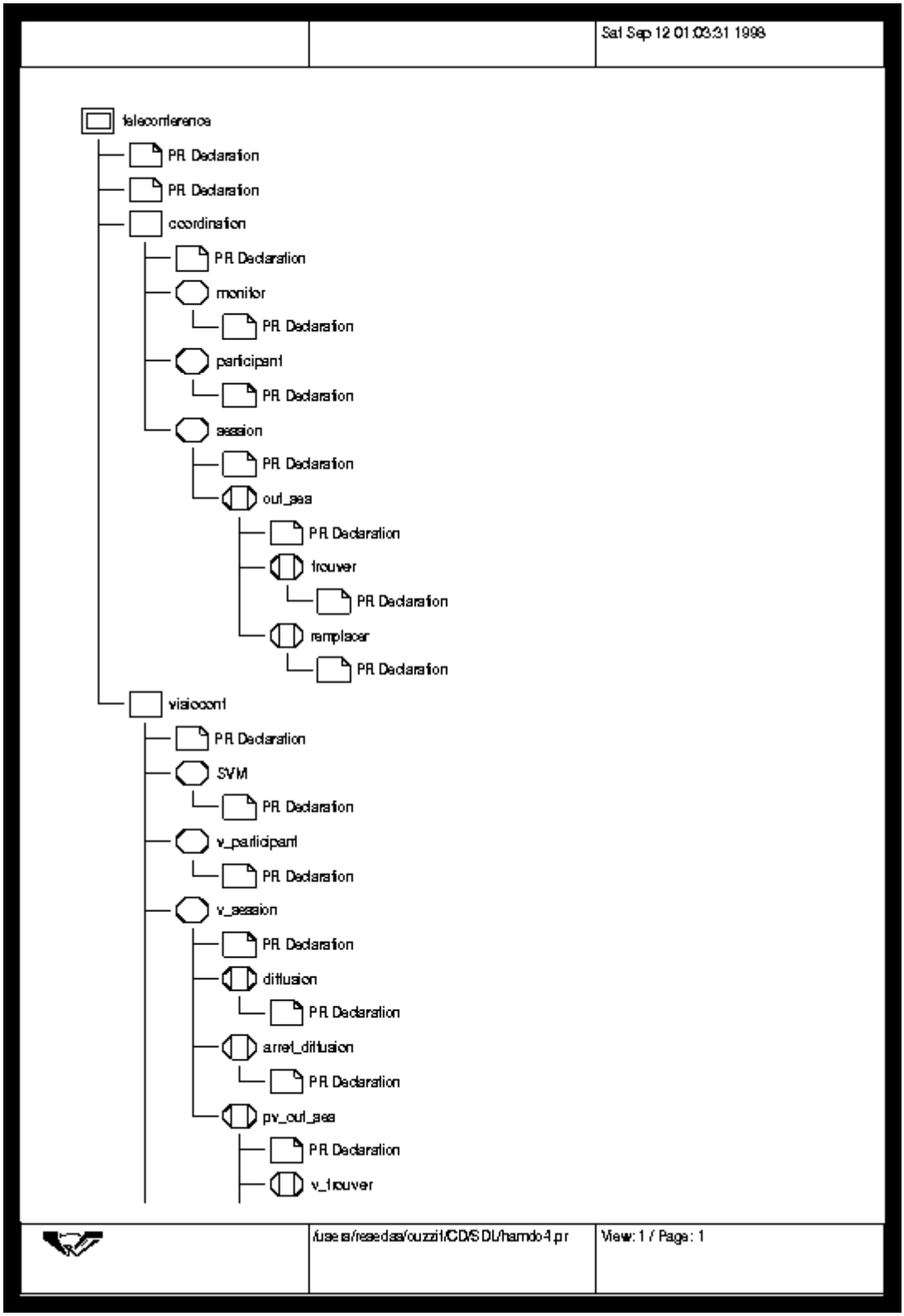
[Owez 97] Philippe Owesarski, Veronique Baudin et Michel Diaz, « Conception et développement d'un système synchrone de télé-formation professionnelle ». Calculateurs parallèles, Volume 9, numéro 2, 1997.

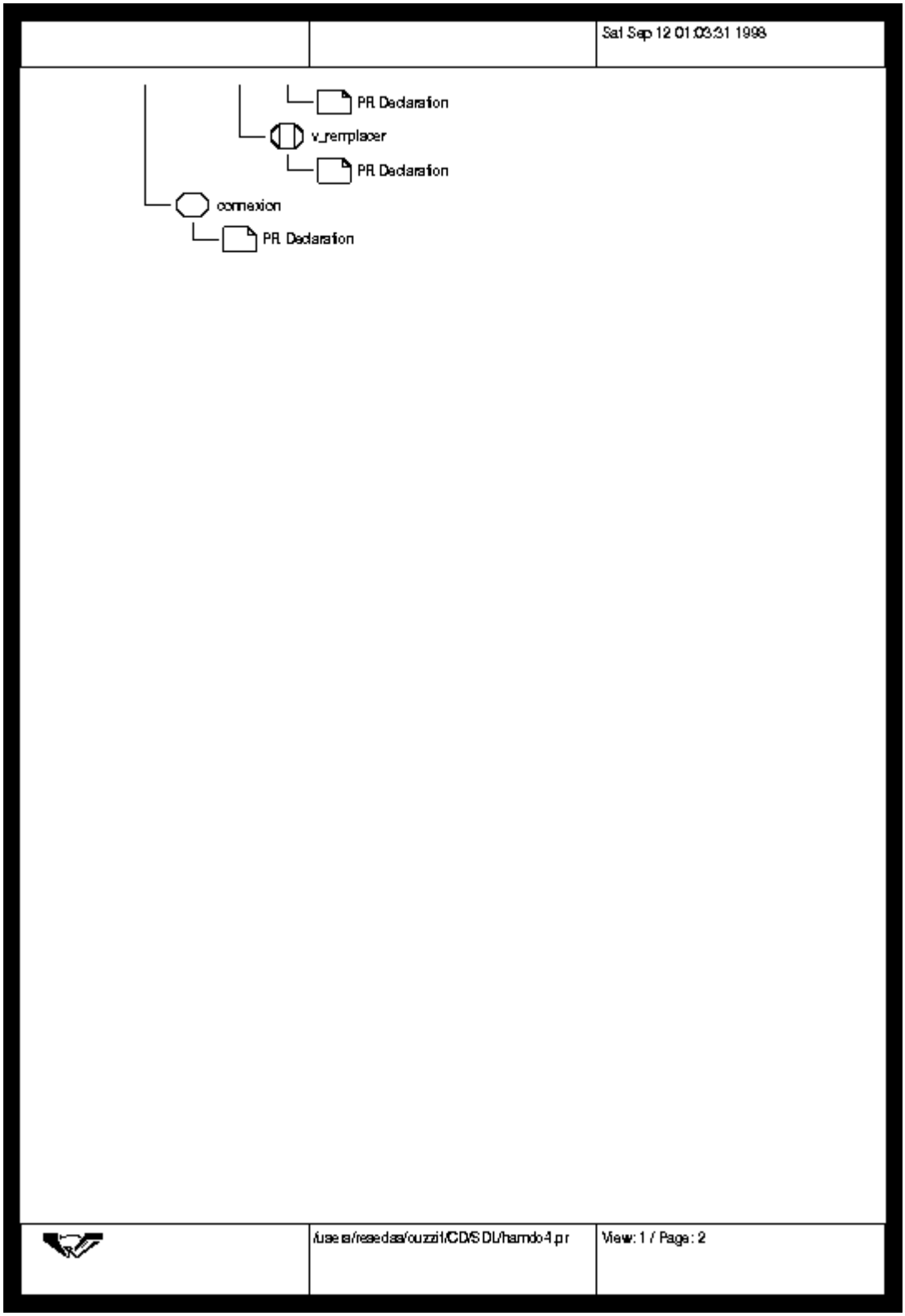
[Sass 94] M.A.Sasse, Claus-Dieter Schulz, Thierry Turletti, «*Remote Seminars through Multimedia Conferencing* » : Experiences from the MICE project, INET/JENC5 1994.

[Turn 88] K.J.Turner, «*Formal Description Techniques* », North-Holland, 1988.

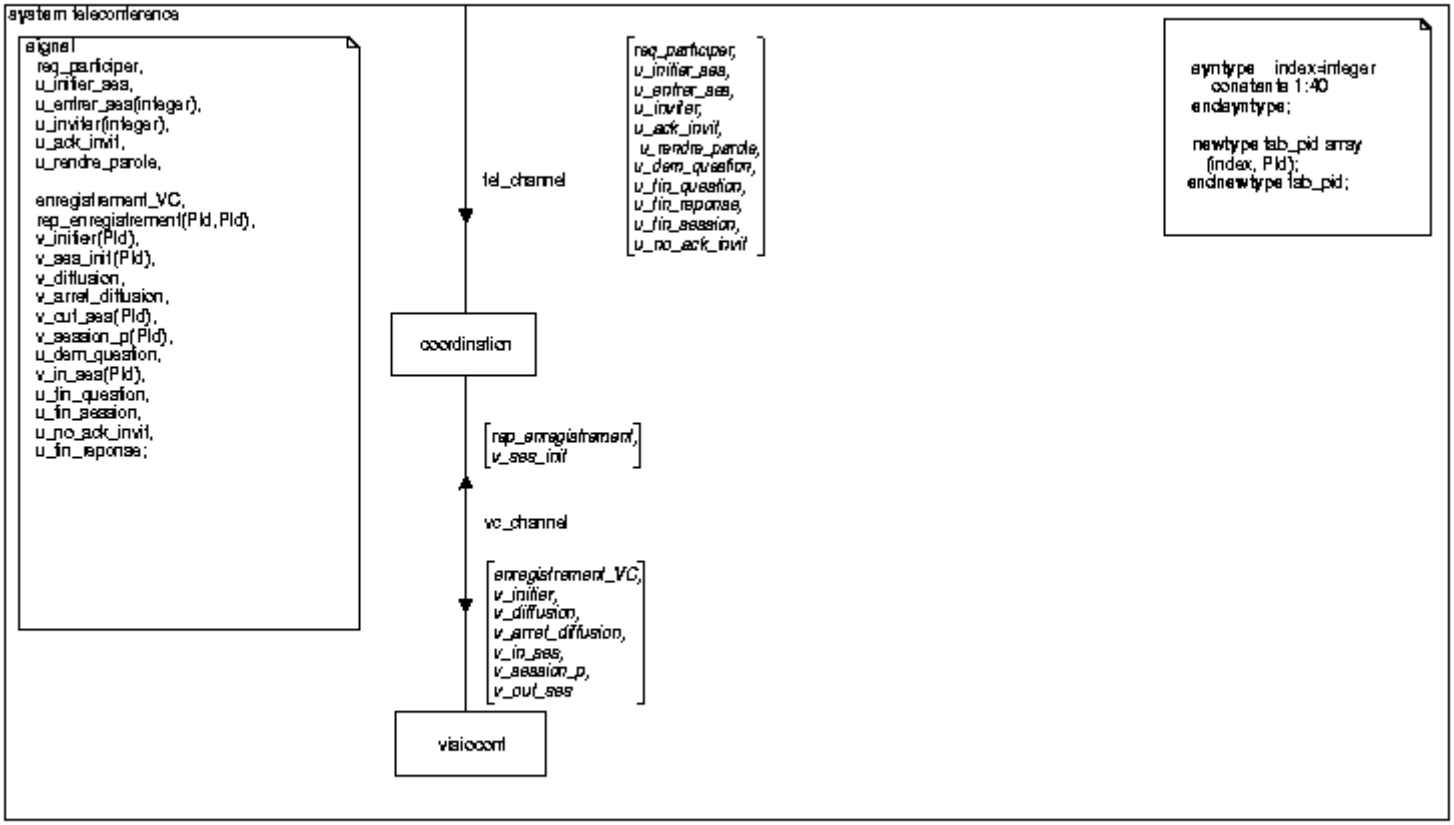
[Turn 93] K.J.Turner, « Using Formal Description Techniques, *« An Introduction to ESTELLE, LOTOS and SDL »* WILEY, 1993.

Annexe


Sat Sep 12 01:09:31 1998
telecontference
PR Declaration
PR Declaration
coordination
PR Declaration
monitor
PR Declaration
participant
PR Declaration
session
PR Declaration
out_sea
PR Declaration
trouver
PR Declaration
remplacer
PR Declaration
visioconf
PR Declaration
SVM
PR Declaration
v_participant
PR Declaration
v_session
PR Declaration
diffusion
PR Declaration
arret_diffusion
PR Declaration
pv_out_sea
PR Declaration
v_trouver
/users/reseclas/ouzzif/CDS/SDL/harndo4.pr
View: 1 / Page: 1

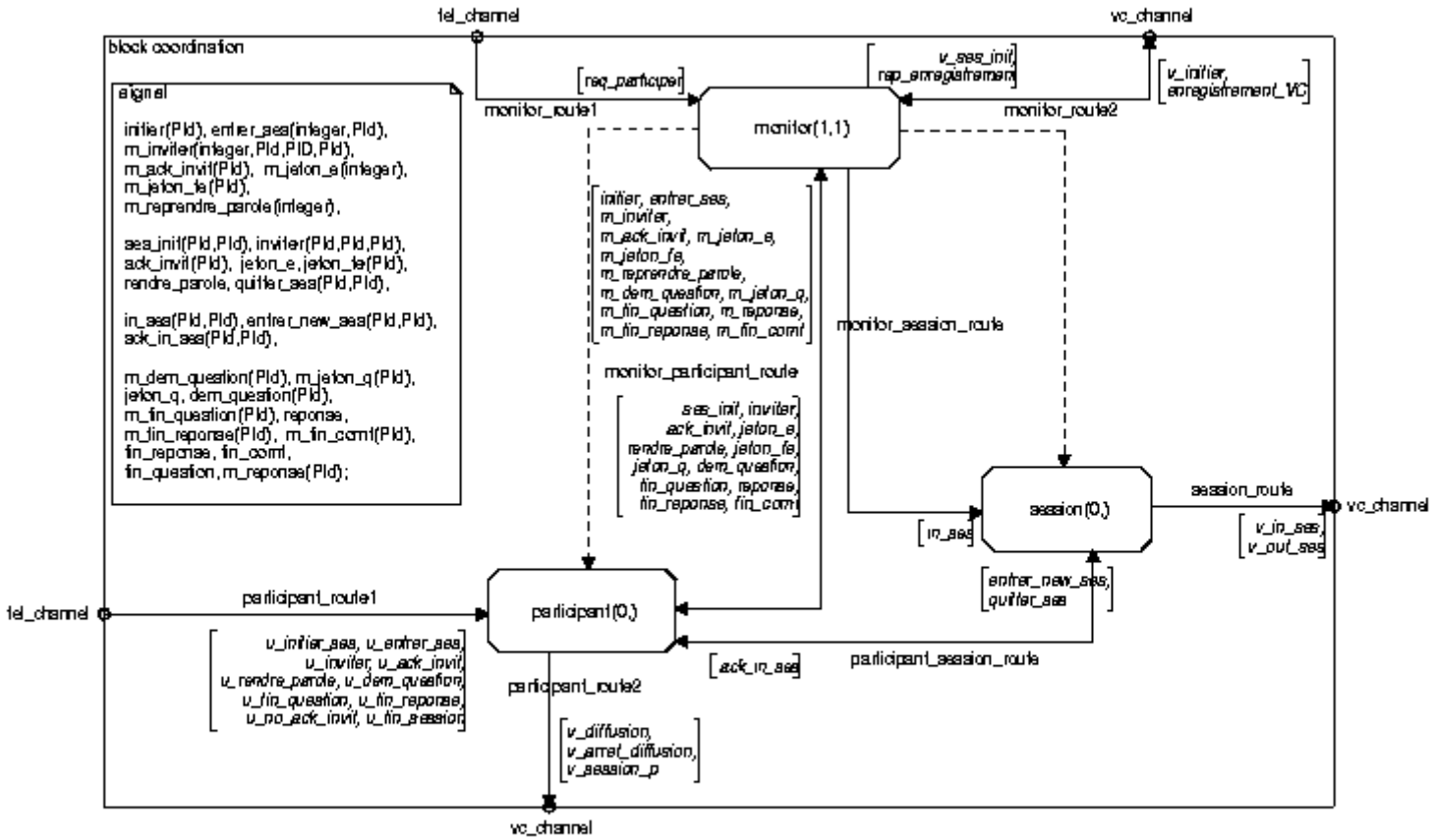

Sat Sep 12 01:09:31 1998
PR Declaration
v_remplacer
PR Declaration
connexion
PR Declaration
/users/reseclas/ouzzif/CDVSDL/harndo4.pr
View: 1 / Page: 2

teleconference		Sat Sep 12 01:03:31 1998

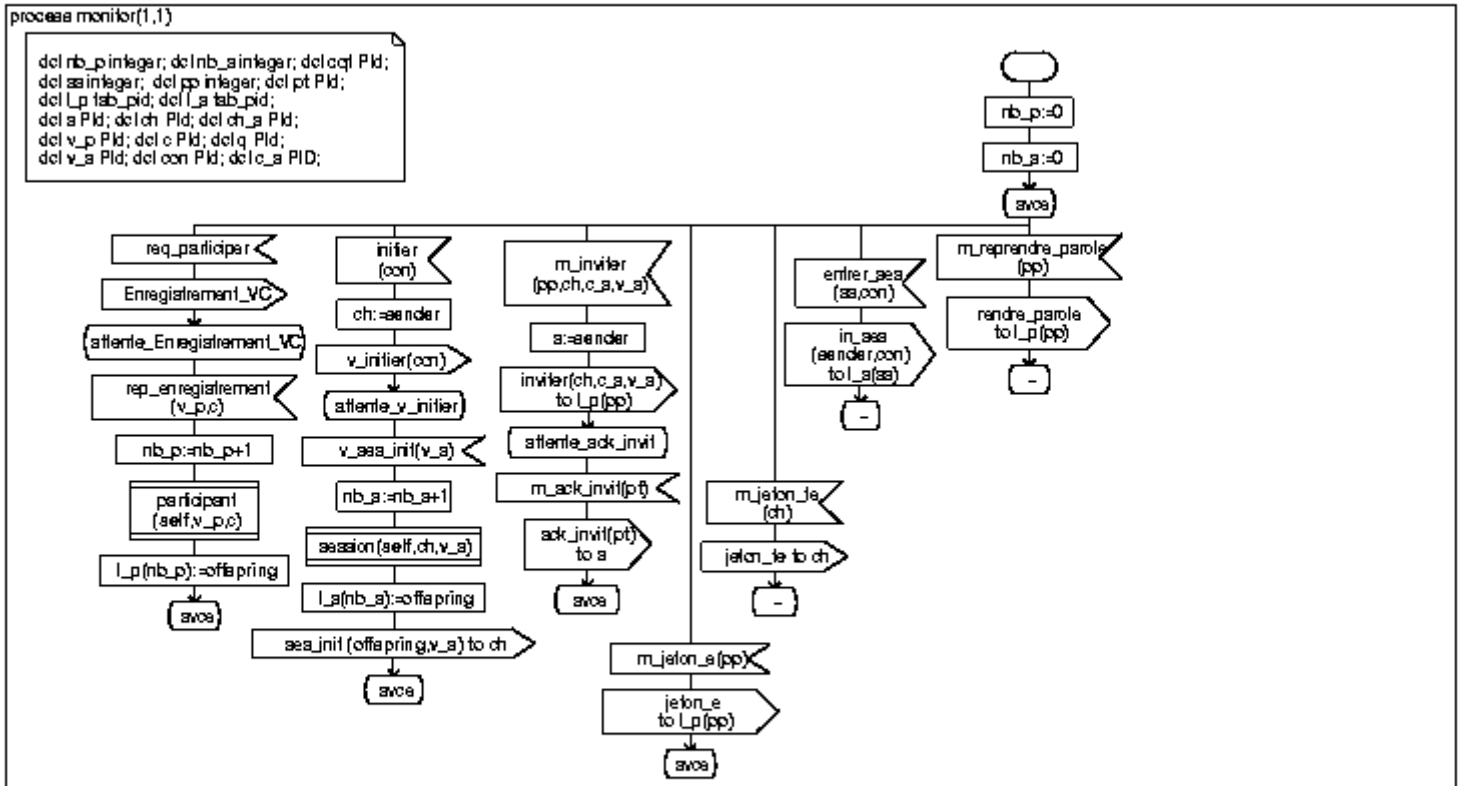


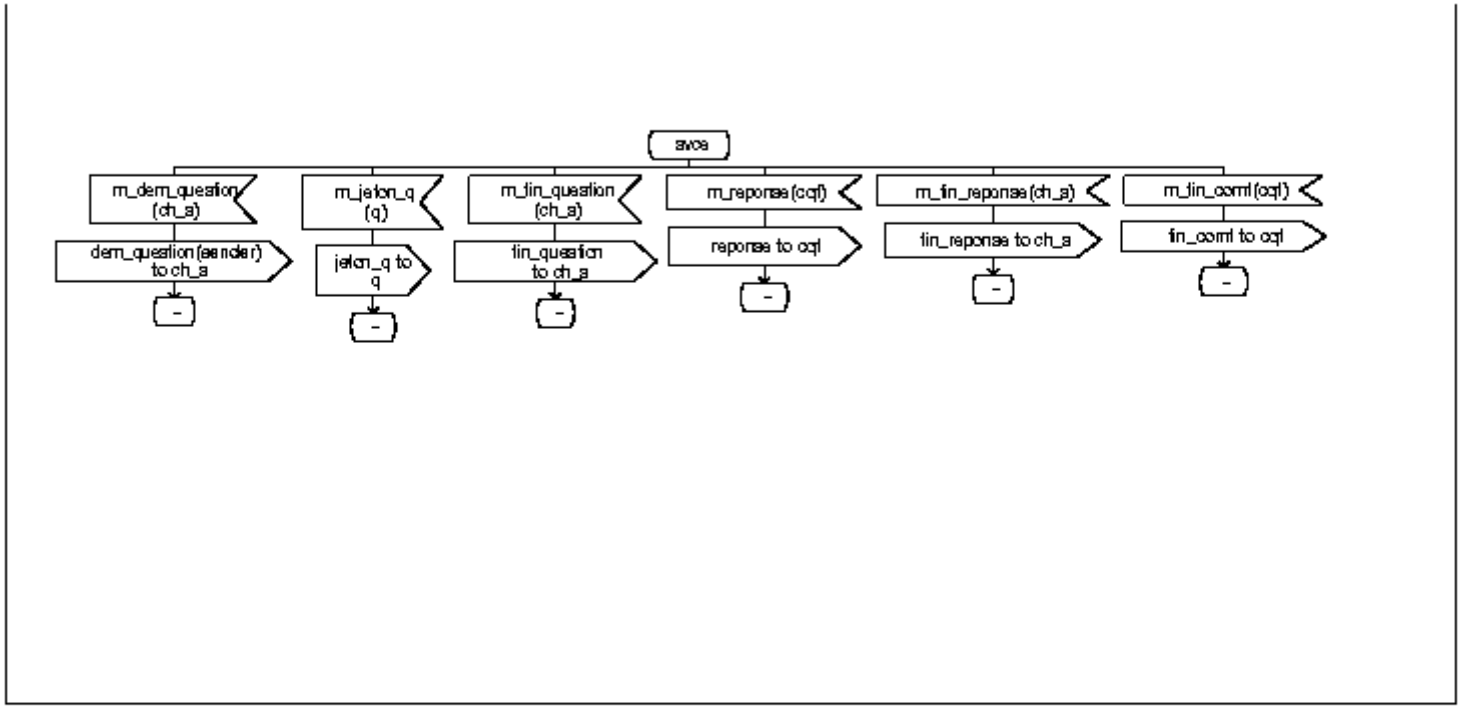
	/users/reseda/bouzid/CD/SDL/hamdo4.pr	View: 2 / Page: 3

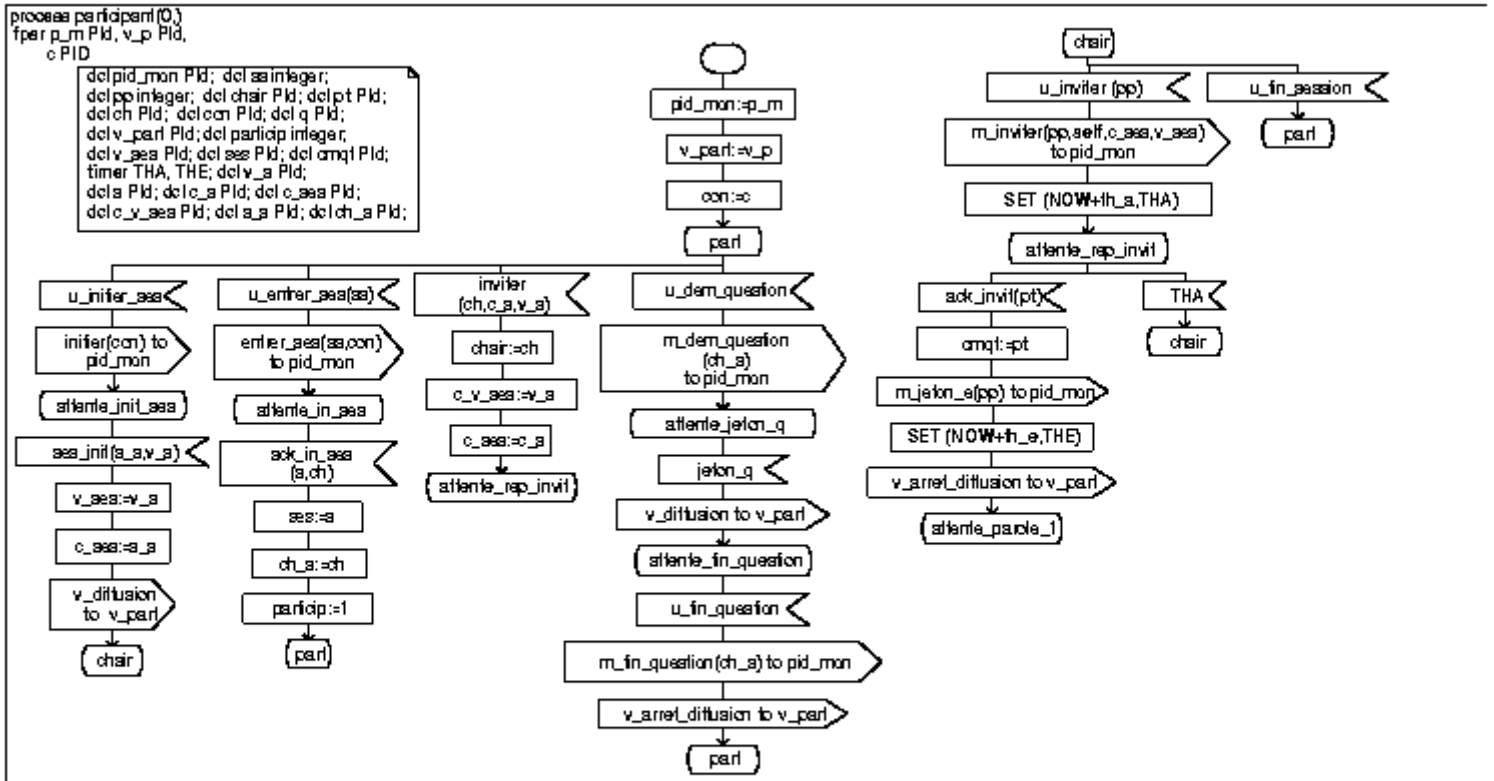
coordination		Sat Sep 12 01:03:31 1998

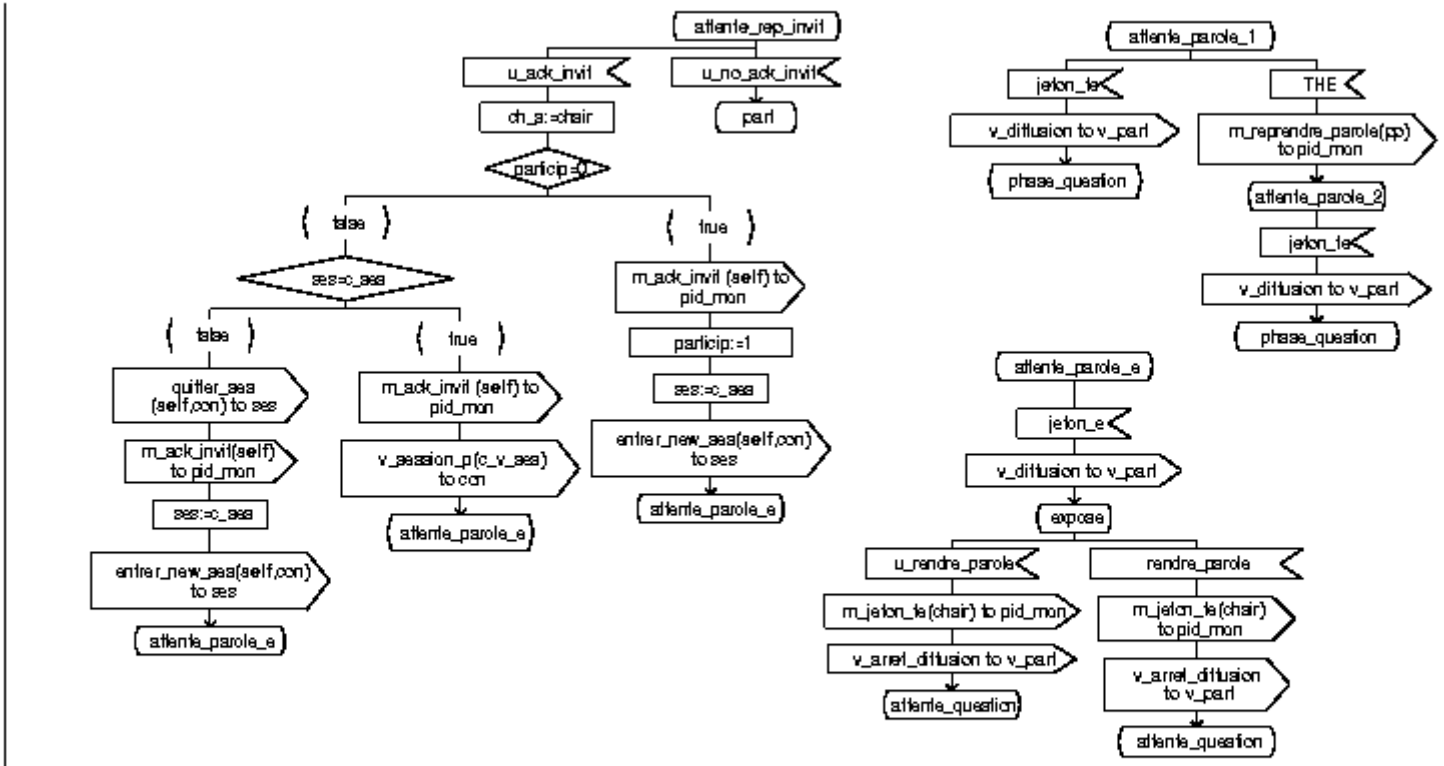


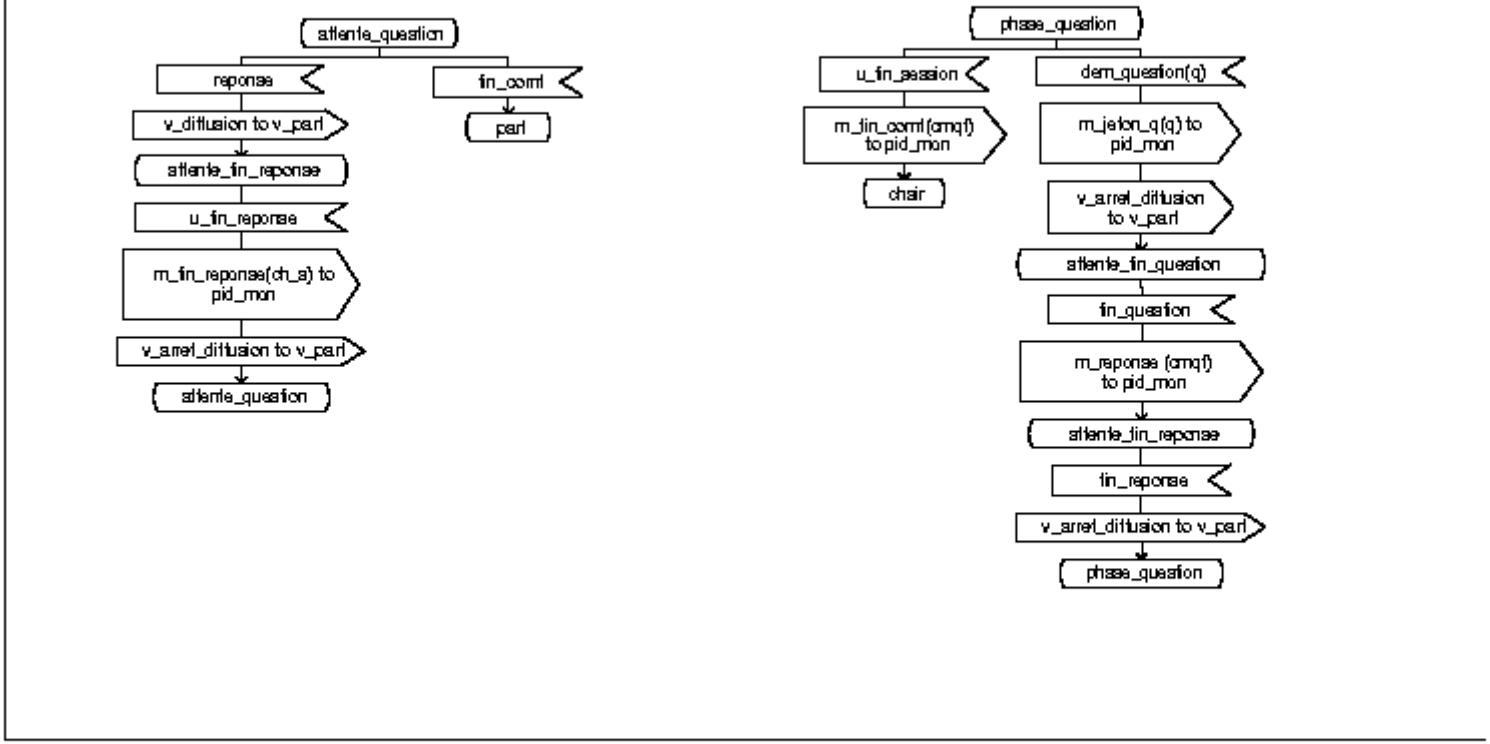
	/users/reseda/ouzzif/CD/SDL/hamdo4.pr	View: 3 / Page: 4

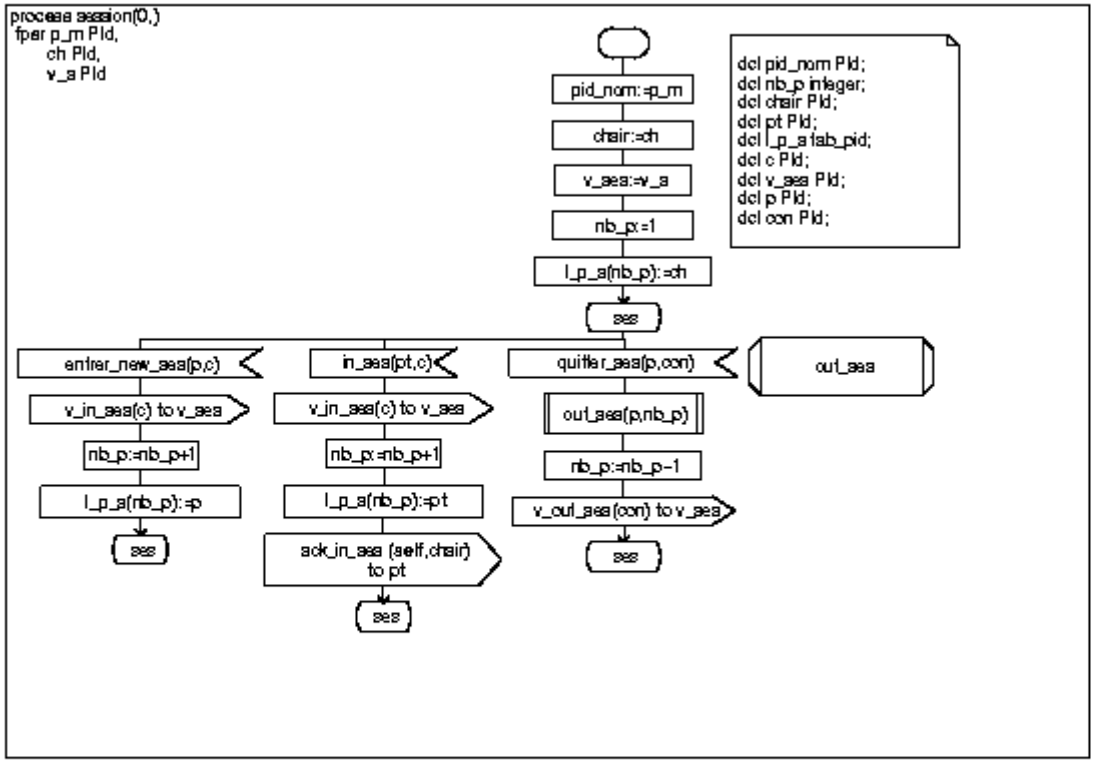

process monitor(1,1)
dcl nb_p integer; dcl nb_a integer; dcl cqt Pid;
dcl aa integer; dcl pp integer; dcl pt Pid;
dcl l_p tab_pid; dcl l_a tab_pid;
dcl a Pid; dcl ch Pid; dcl ch_a Pid;
dcl v_p Pid; dcl c Pid; dcl q Pid;
dcl v_a Pid; dcl con Pid; dcl c_a PID;
nb_p:=0
nb_a:=0
avce
req_participer
Enregistrement_VC
attente_Enregistrement_VC
rep_enregistrement (v_p,c)
nb_p:=nb_p+1
participant (self,v_p,c)
l_p(nb_p):=offspring
avce
initier (con)
ch:=sender
v_initier(con)
attente_v_initier
v_aea_init(v_a)
nb_a:=nb_a+1
session(self,ch,v_a)
l_a(nb_a):=offspring
aea_init (offspring,v_a) to ch
avce
m_inviter (pp,ch,c_a,v_a)
a:=sender
inviter(ch,c_a,v_a) to l_p(pp)
attente_ack_invit
m_ack_invit(pt)
ack_invit(pt) to a
avce
m_jeton_a(pp)
jeton_e to l_p(pp)
avce
entrer_aea (aa,con)
in_aea (sender,con) to l_a(aa)
-
m_jeton_le (ch)
jeton_le to ch
-
m_reprendre_parole (pp)
rendre_parole to l_p(pp)
-


awoa
m_dem_question (ch_a)
dem_question (sender) to ch_a
m_jeton_q (q)
jeton_q to q
m_fin_question (ch_a)
fin_question to ch_a
m_reponse (cqf)
reponse to cqf
m_fin_reponse (ch_a)
fin_reponse to ch_a
m_fin_comf (cqf)
fin_comf to cqf

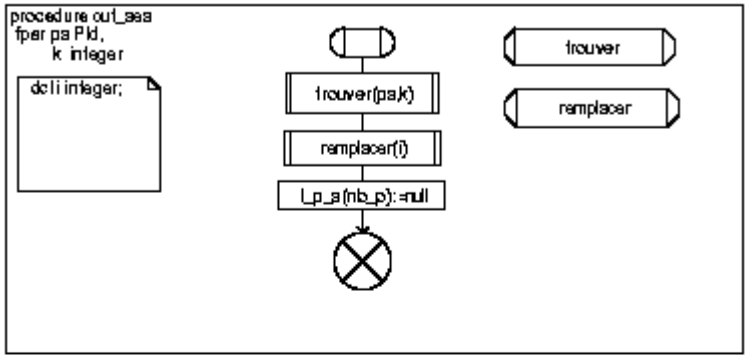

process participant(0,)
fpar p_m Pid, v_p Pid,
c PID
dcl pid_mon Pid; dcl ass integer;
dcl pp integer; dcl chair Pid; dcl pt Pid;
dcl ch Pid; dcl con Pid; dcl q Pid;
dcl v_parl Pid; dcl particip integer;
dcl v_ass Pid; dcl ass Pid; dcl cmpt Pid;
timer THA, THE; dcl v_a Pid;
dcl a Pid; dcl c_a Pid; dcl c_ass Pid;
dcl c_v_ass Pid; dcl a_a Pid; dcl ch_a Pid;
pid_mon:=p_m
v_parl:=v_p
con:=c
parl
u_initier_ass
initier(con) to pid_mon
attente_init_ass
ass_init(a_a,v_a)
v_ass:=v_a
c_ass:=a_a
v_diffusion to v_parl
chair
u_entrer_ass(ass)
entrer_ass(ass,con) to pid_mon
attente_in_ass
ack_in_ass (a,ch)
ass:=a
ch_a:=ch
particip:=1
parl
inviter (ch,c_a,v_a)
chair:=ch
c_v_ass:=v_a
c_ass:=c_a
attente_rep_invit
u_dem_question
m_dem_question (ch_a) to pid_mon
attente_jeton_q
jeton_q
v_diffusion to v_parl
attente_fin_question
u_fin_question
m_fin_question(ch_a) to pid_mon
v_arret_diffusion to v_parl
parl
chair
u_inviter (pp)
u_fin_session
parl
m_inviter(pp,self,c_ass,v_ass) to pid_mon
SET (NOW+th_a,THA)
attente_rep_invit
ack_invit(pt)
THA
chair
cmpt:=pt
m_jeton_e(pp) to pid_mon
SET (NOW+th_e,THE)
v_arret_diffusion to v_parl
attente_parole_1


attente_rep_invit
u_ack_invit
u_no_ack_invit
ch_a:=chair
parl
particip=0
false
true
ses=c_ses
false
true
quitter_ses (self,con) to ses
m_ack_invit(self) to pid_mon
ses:=c_ses
entrer_new_ses(self,con) to ses
attente_parole_e
m_ack_invit (self) to pid_mon
v_session_p(c_v_ses) to con
attente_parole_e
m_ack_invit (self) to pid_mon
particip:=1
ses:=c_ses
entrer_new_ses(self,con) to ses
attente_parole_e
attente_parole_1
jeton_te
THE
v_diffusion to v_parl
phrase_question
m_reprendre_parole(pp) to pid_mon
attente_parole_2
jeton_te
v_diffusion to v_parl
phrase_question
attente_parole_e
jeton_e
v_diffusion to v_parl
exposa
u_rendre_parole
rendre_parole
m_jeton_te(chair) to pid_mon
v_arret_diffusion to v_parl
attente_question
m_jeton_te(chair) to pid_mon
v_arret_diffusion to v_parl
attente_question

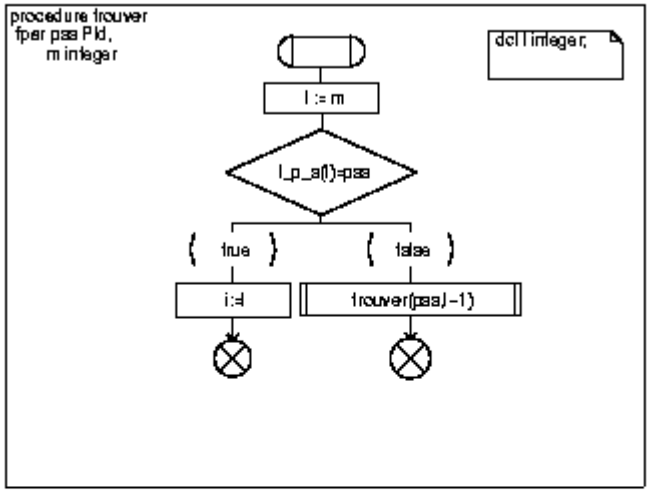

attente_question
reponse
fin_comm
v_diffusion to v_parl
parl
attente_fin_reponse
u_fin_reponse
m_fin_reponse(ch_a) to pid_mon
v_arret_diffusion to v_parl
attente_question
phase_question
u_fin_session
dem_question(q)
m_fin_comm(cmqf) to pid_mon
m_jeton_q(q) to pid_mon
chair
v_arret_diffusion to v_parl
attente_fin_question
fin_question
m_reponse (cmqf) to pid_mon
attente_fin_reponse
fin_reponse
v_arret_diffusion to v_parl
phase_question


process session(0,)
fpar p_m Pid,
ch Pid,
v_a Pid
pid_norm:=p_m
chair:=ch
v_ses:=v_a
nb_p:=1
l_p_a(nb_p):=ch
ses
dcl pid_norm Pid;
dcl nb_p integer;
dcl chair Pid;
dcl pt Pid;
dcl l_p_a fab_pid;
dcl c Pid;
dcl v_ses Pid;
dcl p Pid;
dcl con Pid;
entrer_new_ses(p,c)
v_in_ses(c) to v_ses
nb_p:=nb_p+1
l_p_a(nb_p):=p
ses
in_ses(pt,c)
v_in_ses(c) to v_ses
nb_p:=nb_p+1
l_p_a(nb_p):=pt
ack_in_ses (self,chair) to pt
ses
quitter_ses(p,con)
out_ses(p,nb_p)
nb_p:=nb_p-1
v_out_ses(con) to v_ses
ses
out_ses

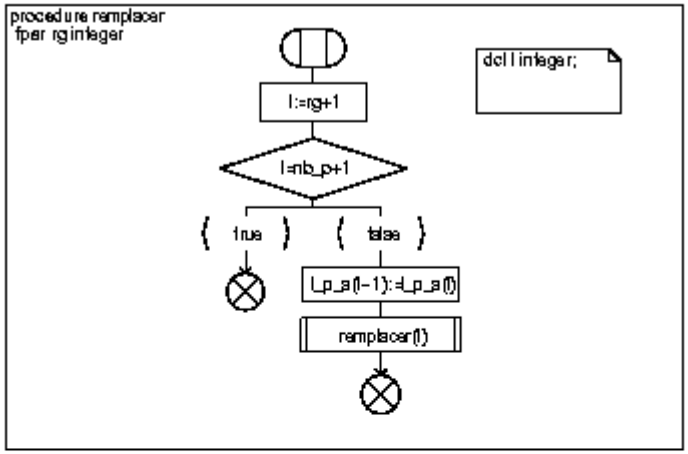
out_aea		Sat Sep 12 01:03:31 1998



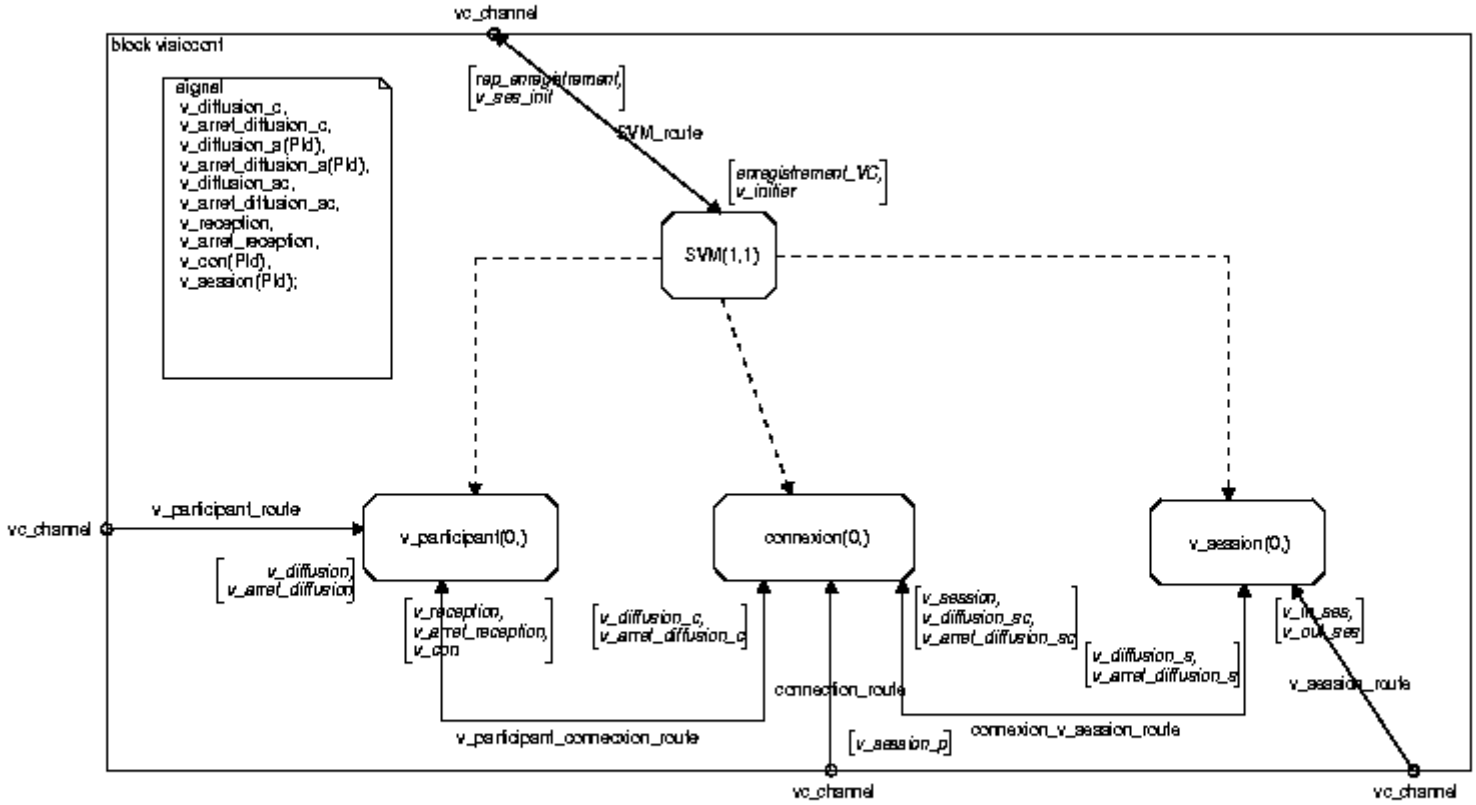
	/users/reseedaa/ouzzif/CD/SDL/hamdo4.pr	View: 7 / Page: 14


procedure trouver
fper pss Pid,
m integer
dcl I integer;
I := m
I_p_s(I)=pss
true
false
i:=I
trouver(pss,I-1)

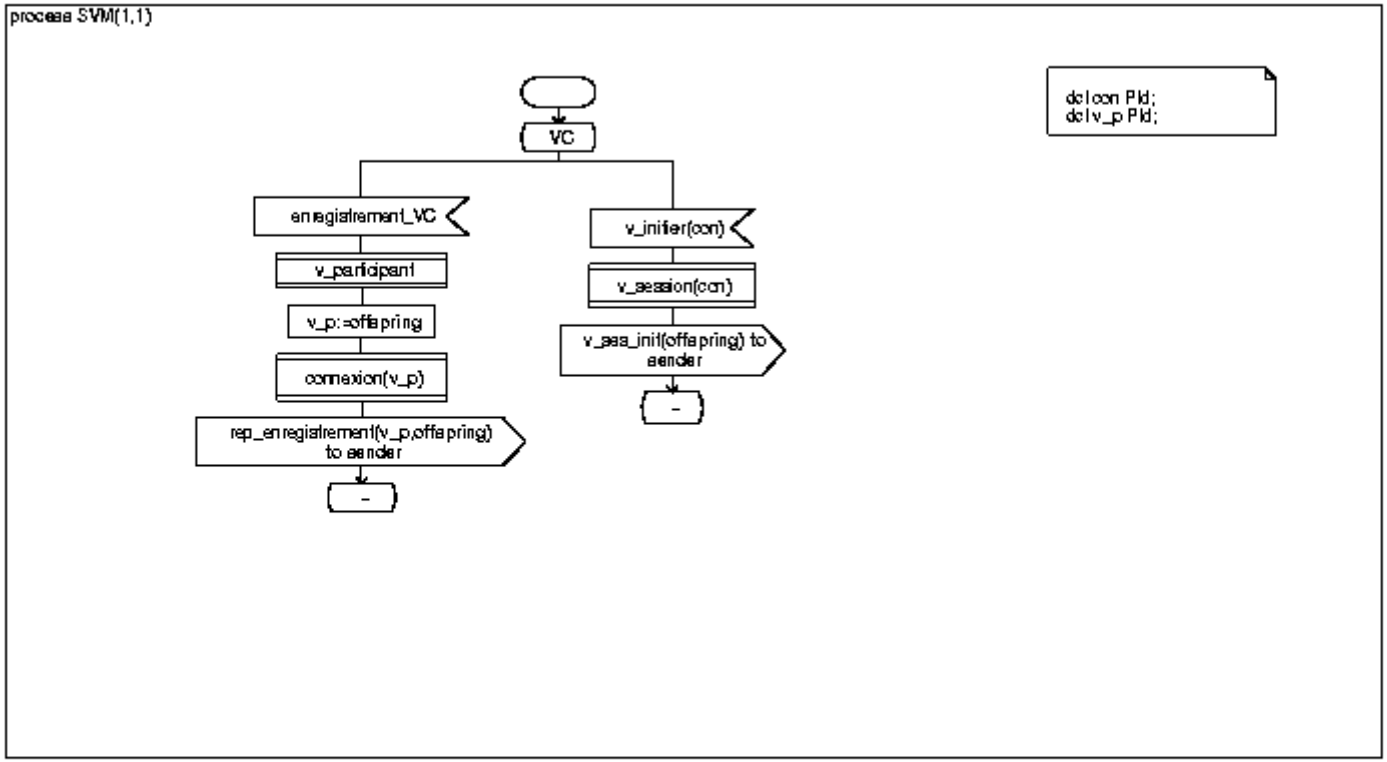
remplacer		Sat Sep 12 01:03:31 1998



	/users/reseda/ouzzif/CD/SDL/hamdo4.pr	View: 9 / Page: 16

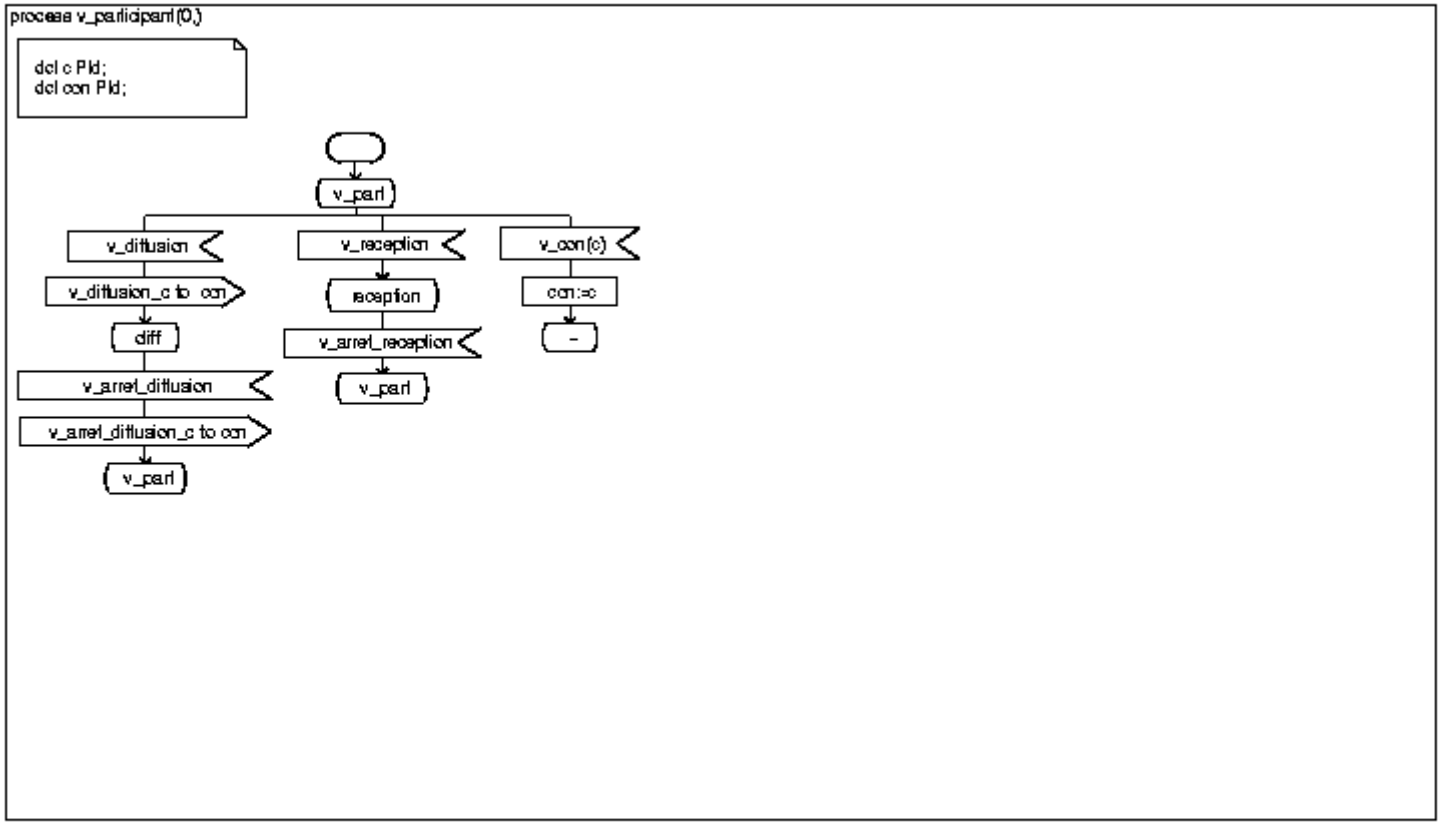

vc_channel
block visiocon1
signal
v_diffusion_c,
v_arret_diffusion_c,
v_diffusion_s(Pid),
v_arret_diffusion_s(Pid),
v_diffusion_sc,
v_arret_diffusion_sc,
v_reception,
v_arret_reception,
v_con(Pid),
v_session(Pid);
rep_enregistrement,
v_ses_init
SVM_route
enregistrement_VC,
v_initier
SVM(1,1)
vc_channel
v_participant_route
v_participant(0,)
connexion(0,)
v_session(0,)
v_diffusion,
v_arret_diffusion
v_reception,
v_arret_reception,
v_con
v_diffusion_c,
v_arret_diffusion_c
v_session,
v_diffusion_sc,
v_arret_diffusion_sc
v_diffusion_s,
v_arret_diffusion_s
v_in_ses,
v_out_ses
connection_route
v_session_route
v_participant_connexion_route
connexion_v_session_route
v_session_p
vc_channel
vc_channel

SVM		Sat Sep 12 01:03:31 1998



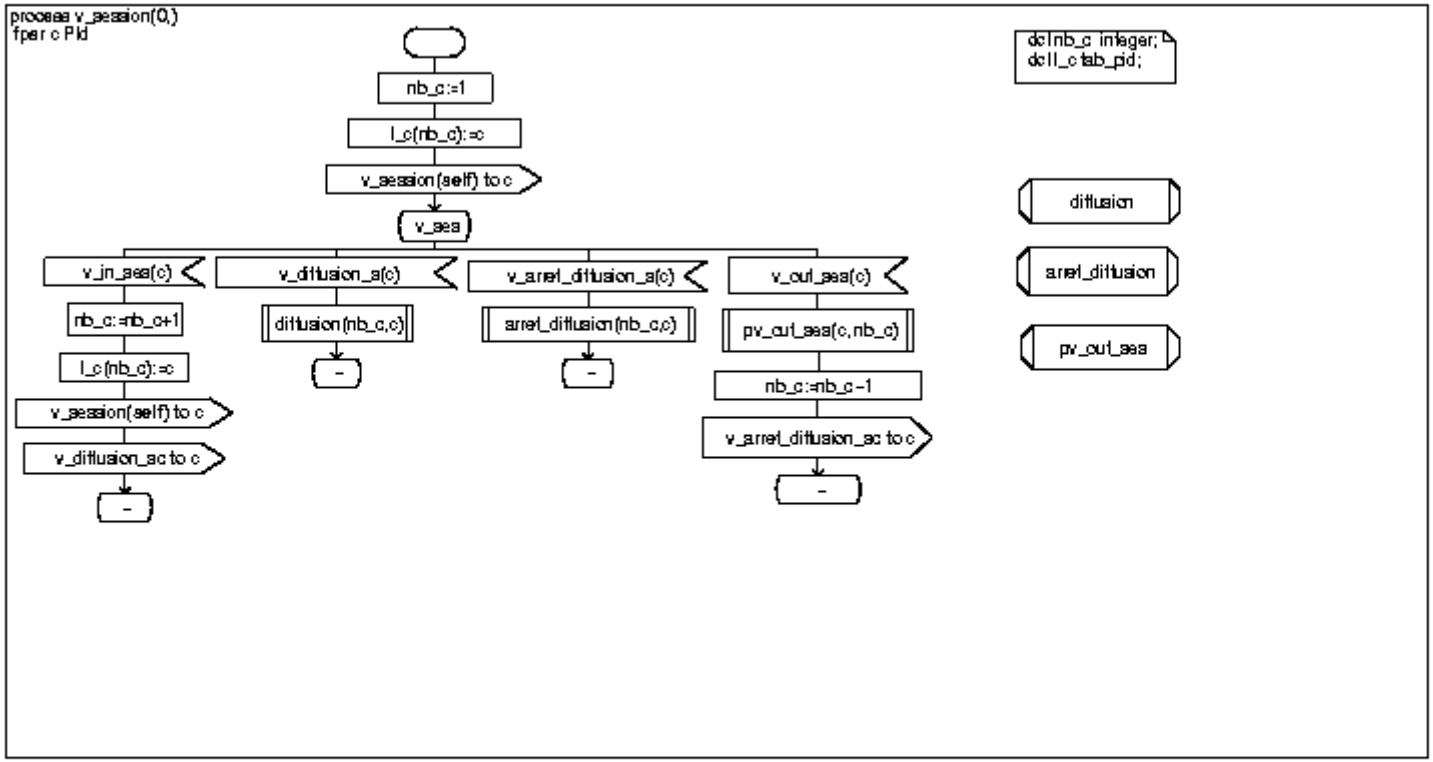
	/users/reseda/bouzid/CD/SDL/hamdo4.pr	View: 11 / Page: 18

v_participant		Sat Sep 12 01:03:31 1998

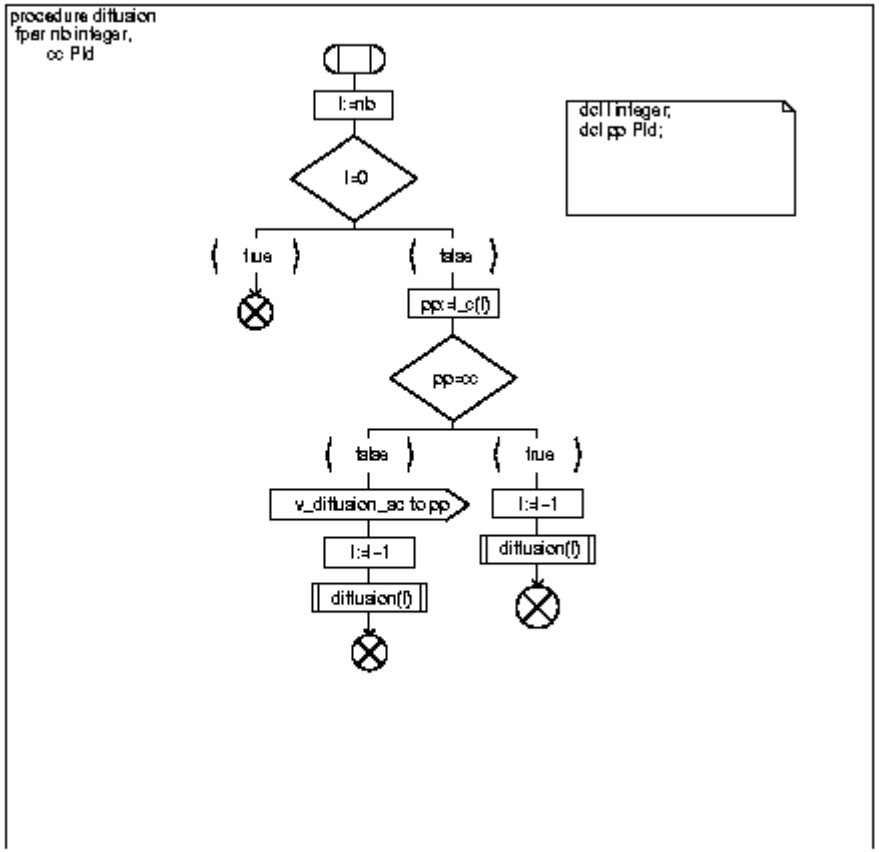


	/users/reseda/bouzid/CD/SDL/hamdo4.pr	View: 12 / Page: 19

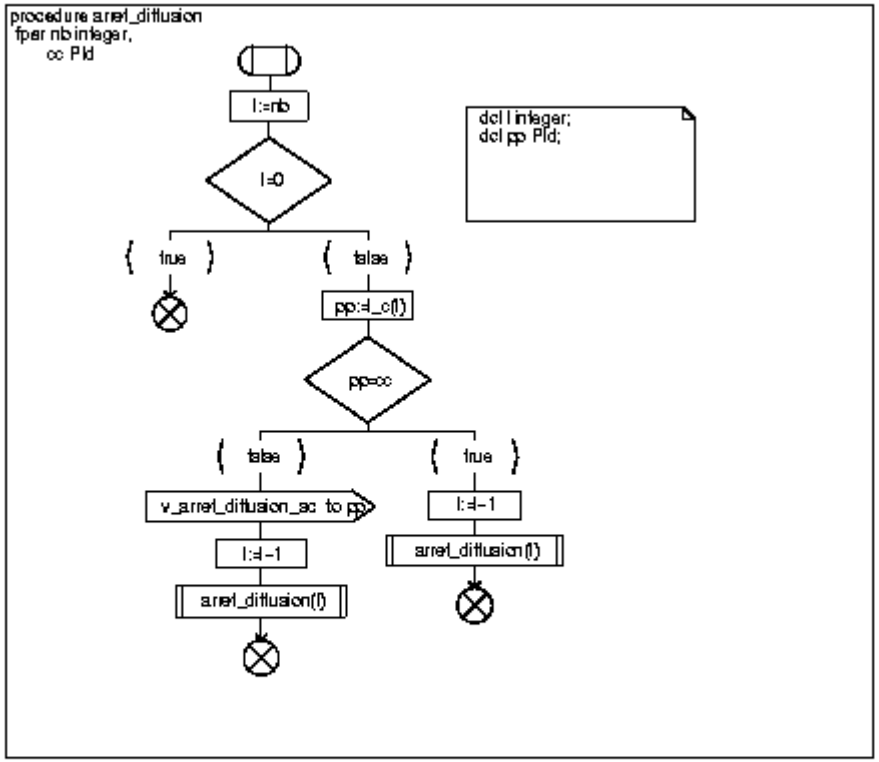
v_session		Sat Sep 12 01:03:31 1998



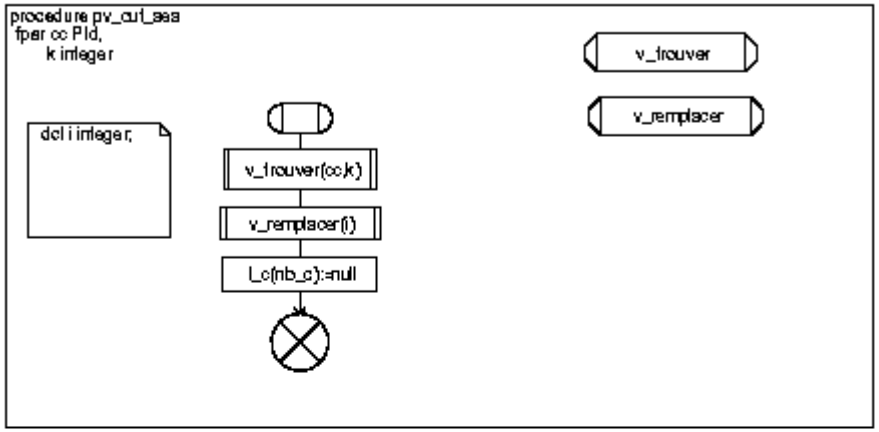
	/users/resedas/ouzzif/CD/SDL/hamdo4.pr	View: 13 / Page: 20


procedure diffusion
fper nb integer,
cc Pid
l:=nb
l=0
true
false
pp:=l_c(l)
pp=cc
false
true
v_diffusion_ac to pp
l:=l-1
diffusion(l)
l:=l-1
diffusion(l)
dcl l integer;
dcl pp Pid;

arret_diffusion		Sat Sep 12 01:03:31 1998

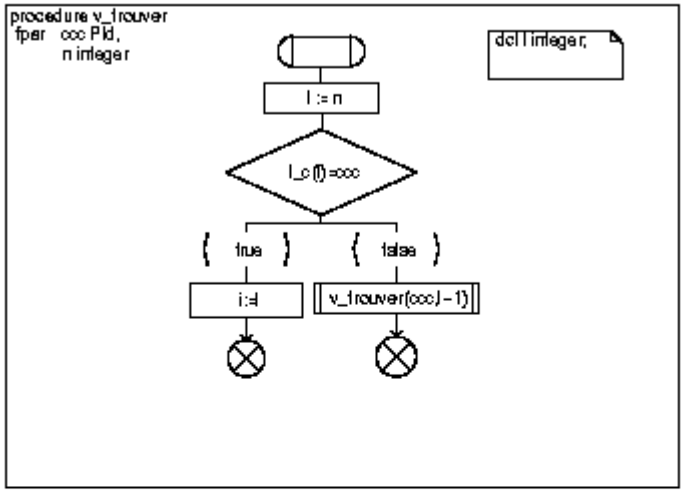


pv_cut_aaa		Sat Sep 12 01:03:31 1998



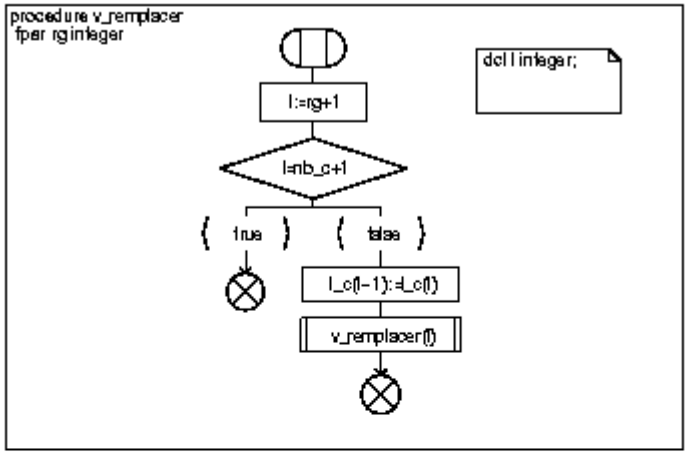
	/users/reseedaa/bouzzit/CD/SDL/hamdo4.pr	View: 16 / Page: 24

v_trouver		Sat Sep 12 01:03:31 1998

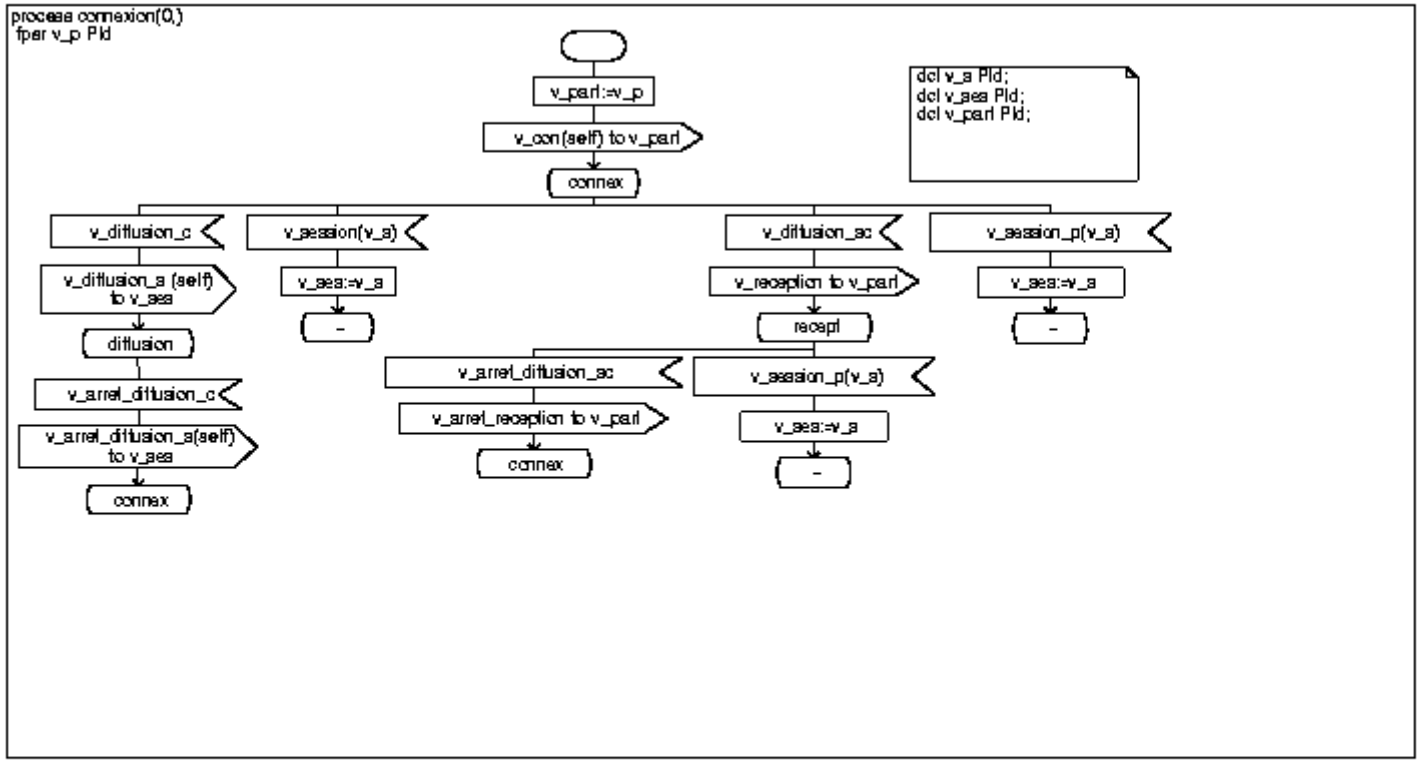


	/users/reseda/bouzzit/CD/SDL/hamdo4.pr	View: 17 / Page: 25

v_remplacer		Sat Sep 12 01:03:31 1998



	/users/reseedaa/bouzzit/CD/SDL/hamdo4.pr	View: 18 / Page: 26


process connexion(0,)
fpar v_p Pid
v_parl:=v_p
v_con(self) to v_parl
connex
dcl v_a Pid;
dcl v_aea Pid;
dcl v_parl Pid;
v_diffusion_c
v_session(v_a)
v_diffusion_ac
v_session_p(v_a)
v_diffusion_a (self) to v_aea
v_aea:=v_a
v_reception to v_parl
v_aea:=v_a
diffusion
recepl
v_arret_diffusion_c
v_arret_diffusion_ac
v_session_p(v_a)
v_arret_diffusion_a(self) to v_aea
v_arret_reception to v_parl
v_aea:=v_a
connex
connex